

DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
<code>project_id</code>	A unique identifier for the proposed project. Example: p
<code>project_title</code>	Title of the project. Examples: <ul style="list-style-type: none">• Art Will Make You Happy!• First Grade Fun
<code>project_grade_category</code>	Grade level of students for which the project is targeted. enumerated values: <ul style="list-style-type: none">• Grades PreK-2• Grades 3-5• Grades 6-8• Grades 9-12

Feature	Description
project_subject_categories	<p>One or more (comma-separated) subject categories for the project. One of the following enumerated list of values:</p> <ul style="list-style-type: none"> • Applied Learning • Care & Hunger • Health & Sports • History & Civics • Literacy & Language • Math & Science • Music & The Arts • Special Needs • Warmth <p>Examples:</p> <ul style="list-style-type: none"> • Music & The Arts • Literacy & Language, Math & Science
school_state	<p>State where school is located (<u>Two-letter U.S. postal code</u> (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations))</p> <p>Example: WY</p>
project_subject_subcategories	<p>One or more (comma-separated) subject subcategories</p> <p>Examples:</p> <ul style="list-style-type: none"> • Literacy • Literature & Writing, Social Sciences
project_resource_summary	<p>An explanation of the resources needed for the project. It can be a list of resources, a paragraph, or a sentence.</p> <ul style="list-style-type: none"> • My students need hands on literacy materials and sensory needs!
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. Example: 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	<p>Teacher's title. One of the following enumerated values:</p> <ul style="list-style-type: none"> • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.

Feature	Description
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the teacher. Example: 2

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
id	A project_id value from the <code>train.csv</code> file. Example: p036502
description	Description of the resource. Example: Tenor Saxophone Reeds, Box of 25
quantity	Quantity of the resource required. Example: 3
price	Price of the resource required. Example: 9.95

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in `train.csv`, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
project_is_approved	A binary flag indicating whether DonorsChoose approved the project. A value of 0 indicates the project was not approved, and a value of 1 indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with `project_submitted_datetime` of 2016-05-17 and later, the values of `project_essay_3` and `project_essay_4` will be NaN.

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

Output hidden; open in <https://colab.research.google.com> (<https://colab.research.google.com>) to view.

In [2]:

```
import pickle
# Load the Drive helper and mount
from google.colab import drive
drive.mount('/content/drive')
```

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:

.....

Mounted at /content/drive

In [3]:

```
dir_path = '/content/drive/My Drive/appliedai/Data'
!ls /content/drive/My\ Drive/appliedai
```

Data	Dumps	Logistic_Regression	NB_dumps	SGD	tsne
Decision_Trees	KNN	Naive_Bayes	New_dumps	SVM	

1.1 Reading Data

In [0]:

```
project_data = pd.read_csv(os.path.join(dir_path, 'train_data.csv'))
resource_data = pd.read_csv(os.path.join(dir_path, 'resources.csv'))
print(project_data.shape)
print(resource_data.shape)
```

(109248, 17)

(1541272, 4)

In [0]:

```
print(len(project_data[project_data['project_is_approved'] == 0]))
print(len(project_data[project_data['project_is_approved'] == 1]))
```

16542

92706

In [0]:

```
print("Number of data points in train data", project_data.shape)
print('- '*50)
print("The attributes of data :", project_data.columns.values)
```

Number of data points in train data (109248, 17)

The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
'project_submitted_datetime' 'project_grade_category'
'project_subject_categories' 'project_subject_subcategories'
'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
'project_essay_4' 'project_resource_summary'
'teacher_number_of_previously_posted_projects' 'project_is_approved']

In [0]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

Number of data points in train data (1541272, 4)

['id' 'description' 'quantity' 'price']

Out[16]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of project_subject_categories

In [0]:

```
categories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the category based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
            j = j.replace(' ', '') # we are replacing all the ' '(space) with ''(empty) ex:"Math
            temp+=j.strip()+" " # " abc ".strip() will return "abc", remove the trailing spaces
            temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of project_subject_subcategories

In [0]:

```

sub_categories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the category based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ', '') # we are placing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" #" " abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))

```

1.4 Text preprocessing

1.4.1 Essay Text

In [0]:

```

# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) + \
    project_data["project_essay_2"].map(str) + \
    project_data["project_essay_3"].map(str) + \
    project_data["project_essay_4"].map(str)

```


In [0]:

```
project_data.head(2)
```

Out[20]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_stat
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

In [0]:

```
type(project_data['project_is_approved'][0])
```

Out[21]:

numpy.int64

In [0]:

```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school. Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect. "The limits of your language are the limits of your world." -Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All families with students within the Level 1 proficiency status, will be offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\nnnannan

=====

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity. My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the st

ools who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan

=====

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an "open classroom" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more. With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs a lot of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

=====

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

=====

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character. In our classroom we can utilize

lize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan

=====

In [0]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can't", "can not", phrase)

    # general
    phrase = re.sub(r"n't", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [0]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

=====

In [0]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python
sent = sent.replace('\r', ' ')
sent = sent.replace('\n', ' ')
sent = sent.replace('\t', ' ')
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. The materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. They also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves. nannan

In [0]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays cognitive delays gross fine motor delays to autism They are eager beavers and always strive to work their hardest working past their limitations The materials we have are the ones I seek out for my students I teach in a Title I school where most of the students receive free or reduced price lunch Despite their disabilities and limitations my students love coming to school and come eager to learn and explore Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting This is how my kids feel all the time They want to be able to move as they learn or so they say Wobble chairs are the answer and I love them because they develop their core which enhances gross motor and in turn fine motor skills They also want to learn through games my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Physical engagement is the key to our success The number toss and color and shape mats can make that happen My students will forget they are doing work and just have the fun a 6 year old deserves nannan

In [0]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "
    "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
    'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they',
    'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
    'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
    'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
    'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'each',
    'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
    's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
    've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do',
    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
    'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:

```
# Combining all the above statements
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\r', ' ')
    sent = sent.replace('\n', ' ')
    sent = sent.replace('\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

100%|██████████| 109248/109248 [01:06<00:00, 1630.63it/s]

In [0]:

```
# after preprocessing
preprocessed_essays[20000]
cleaned_project_data = pd.DataFrame()
# project_data['cleaned_essay'] = preprocessed_essays
cleaned_project_data['id'] = project_data['id']
cleaned_project_data['cleaned_essay'] = preprocessed_essays
cleaned_project_data.head(2)
```

Out[29]:

	id	cleaned_essay
0	p253737	my students english learners working english s...
1	p258326	our students arrive school eager learn they po...

1.4.2 Project title Text

In [0]:

```
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
=====
More Movement with Hokki Stools
=====
Sailing Into a Super 4th Grade Year
=====
We Need To Move It While We Input It!
=====
Inspiring Minds by Enhancing the Educational Experience
=====
```

In [0]:

```
# similarly you can preprocess the titles also
preprocessed_titles = []
for sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:00, 34012.09it/s]
```

In [0]:

```
# similarly you can preprocess the project grade also
preprocessed_grades = []
for sentence in tqdm(project_data['project_grade_category'].values):
    # sent = decontracted(sentence)
    # sent = sent.replace('\\r', ' ')
    # sent = sent.replace('\\\"', ' ')
    # sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', '_', sentence)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_grades.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:00<00:00, 164852.77it/s]
```


In [0]:

```
cleaned_project_data.head(2)
```

Out[37]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategories
0	p253737	my students english learners working english s...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

In [0]:

```
cleaned_project_data['cleaned_text'] = cleaned_project_data['cleaned_essay'].map(str) + \
    cleaned_project_data['cleaned_project_title'].map(str)
cleaned_project_data.head(2)
```

Out[38]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategories
0	p253737	my students english learners working english s...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

In [0]:

```
set(cleaned_project_data['teacher_prefix'])
```

Out[39]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

1.4.3 Vectorizing Numerical features

In [0]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
cleaned_project_data = pd.merge(cleaned_project_data, price_data, on='id', how='left')
```

In [0]:

```
cleaned_project_data['words_in_project_title'] = cleaned_project_data.apply(lambda row: len(row['words_in_project_title']), axis=1)
cleaned_project_data['words_in_essays'] = cleaned_project_data.apply(lambda row: len(row['words_in_essays']), axis=1)
```

In [0]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
negative = []
positive = []
neutral = []
compound = []
for text in tqdm(cleaned_project_data['cleaned_essay']):
    ss = sid.polarity_scores(text)
    negative.append(ss['neg'])
    positive.append(ss['pos'])
    neutral.append(ss['neu'])
    compound.append(ss['compound'])
```

100%|██████████| 109248/109248 [04:30<00:00, 403.24it/s]

In [0]:

```
cleaned_project_data['negative'] = negative
cleaned_project_data['positive'] = positive
cleaned_project_data['neutral'] = neutral
cleaned_project_data['compound'] = compound
```

In [0]:

```
cleaned_project_data.head(2)
```

Out[46]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategories
0	p253737	my students english learners working english S...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

In [0]:

```
cleaned_project_data.to_csv("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/DonorsChoose.csv")
```

In [0]:

```
# check this one: https://www.youtube.com/watch?v=0H0qOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScaler.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329. ... 399.
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standard deviation
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

In [0]:

```
price_standardized
```

Tasks: Decision Tree

1. Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets

- Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
- Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
- Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
- Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)




2. Hyper parameter tuning (best depth in range [1, 5, 10, 50, 100, 500, 100], and the best min_samples_split in range [5, 10, 100, 500])

- Find the best hyper parameter which will give the maximum AUC (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/>) value
- Find the best hyper parameter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. Graphviz

- Visualize your decision tree with Graphviz. It helps you to understand how a decision is being made, given a new vector.
- Since feature names are not obtained from word2vec related models, visualize only BOW & TFIDF decision trees using Graphviz
- Make sure to print the words in each node of the decision tree instead of printing its index.
- Just for visualization purpose, limit max_depth to 2 or 3 and either embed the generated images of graphviz in your notebook, or directly upload them as .png files.

4. Representation of results

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure

- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.

- Along with plotting ROC curve, you need to print the confusion matrix (<https://www.applidaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tp-r-fpr-fnr-tnr-1/>), with predicted and original labels of test data points

- Once after you plot the confusion matrix with the test data, get all the false positive data points
 - Plot the WordCloud WordCloud (<https://www.geeksforgeeks.org/generating-word-cloud-python/>)
 - Plot the box plot with the price of these false positive data points
 - Plot the pdf with the teacher_number_of_previously_posted_projects of these false positive data points

5. [Task-2]

- Select 5k best features from features of Set 2 using feature_importances_ (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>), discard all the other remaining features and then apply any of the model of your choice i.e. (Decision tree, Logistic Regression, Linear SVM), you need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3

6. Conclusion

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link (<http://zetcode.com/python/prettytable/>).



2. Decision Tree

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [0]:

```
# Taking complete data
project_65k = cleaned_project_data[:65000].copy()
project_65k.shape
```

Out[48]:

```
(65000, 19)
```

In [0]:

```
set(project_65k['teacher_prefix'])
```

Out[49]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

In [0]:

```
print(len(project_65k[project_65k['project_is_approved'] == 0]))
print(len(project_65k[project_65k['project_is_approved'] == 1]))
```

```
9895
```

```
55105
```

In [0]:

```
X = project_65k.drop(['project_is_approved'], axis=1)
y = project_65k['project_is_approved']
print(type(y))
```

```
<class 'pandas.core.series.Series'>
```

In [0]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [0]:

```
print(y_train.value_counts())
print(len(y_train == 0))
print(len(y_train == 1))
```

```
1    24736
```

```
0    4442
```

```
Name: project_is_approved, dtype: int64
```

```
29178
```

```
29178
```

In [0]:

```
test = pd.concat([X_train, y_train], axis=1)
test.head(2)
print(len(test[test['project_is_approved'] == 0]))
print(len(test[test['project_is_approved'] == 1]))
```

```
4442
24736
```

In [0]:

```
from sklearn.utils import resample
```

In [0]:

```
#https://elitedatascience.com/imbalanced-classes
# Separate majority and minority classes
project_majority = test[test.project_is_approved==1]
project_minority = test[test.project_is_approved==0]

# Upsample minority class
project_minority_upsampled = resample(project_minority,
                                      replace=True,      # sample with replacement
                                      n_samples=24736,    # to match majority class
                                      random_state=123) # reproducible results

# Combine majority class with upsampled minority class
project_upsampled = pd.concat([project_majority, project_minority_upsampled])

# Display new class counts
project_upsampled.project_is_approved.value_counts()
```

Out[56]:

```
1    24736
0    24736
Name: project_is_approved, dtype: int64
```

In [0]:

```
X_train = project_upsampled.drop(['project_is_approved'], axis=1)
y_train = project_upsampled['project_is_approved']
```

In [0]:

```
set(X_train['teacher_prefix'])
```

Out[58]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

In [0]:

```
print(y_train.value_counts())
```

```
1    24736
0    24736
Name: project_is_approved, dtype: int64
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train.pkl","wb") as file:
    pickle.dump(X_train, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv.pkl","wb") as file:
    pickle.dump(X_cv, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test.pkl","wb") as file:
    pickle.dump(X_test, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_train.pkl","wb") as file:
    pickle.dump(y_train, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_cv.pkl","wb") as file:
    pickle.dump(y_cv, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_test.pkl","wb") as file:
    pickle.dump(y_test, file)
```

In [0]:

```
print(len(X_train), len(y_train))
print(len(X_cv), len(y_cv))
print(len(X_test), len(y_test))
```

```
49472 49472
14372 14372
21450 21450
```

In [0]:

```
print(set(X_train['teacher_prefix'].values))
print(set(X_cv['teacher_prefix'].values))
print(set(X_test['teacher_prefix'].values))
```

```
{'mrs', 'teacher', 'nan', 'ms', 'dr', 'mr'}
{'mrs', 'teacher', 'nan', 'ms', 'dr', 'mr'}
{'mrs', 'teacher', 'nan', 'ms', 'dr', 'mr'}
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Project Category

In [0]:

```
# we use count vectorizer to convert the values into one hot encoded features
# Project Category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binarize=True)
X_train_one_hot_clean_cat = vectorizer.fit_transform(X_train['clean_categories'].values)
X_cv_one_hot_clean_cat = vectorizer.transform(X_cv['clean_categories'].values)
X_test_one_hot_clean_cat = vectorizer.transform(X_test['clean_categories'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encoding ", X_train_one_hot_clean_cat.shape)
print("Shape of matrix after one hot encoding ", X_cv_one_hot_clean_cat.shape)
print("Shape of matrix after one hot encoding ", X_test_one_hot_clean_cat.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding (49472, 9)
Shape of matrix after one hot encoding (14372, 9)
Shape of matrix after one hot encoding (21450, 9)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_one_hot_clean_cat.pkl", "wb") as file:
    pickle.dump(X_train_one_hot_clean_cat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_one_hot_clean_cat.pkl", "wb") as file:
    pickle.dump(X_cv_one_hot_clean_cat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_one_hot_clean_cat.pkl", "wb") as file:
    pickle.dump(X_test_one_hot_clean_cat, file)
```

Project Sub-category

In [0]:

```
# Project Sub-category
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
X_train_one_hot_clean_sub_cat = vectorizer.fit_transform(X_train['clean_subcategories'].values)
X_cv_one_hot_clean_sub_cat = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_one_hot_clean_sub_cat = vectorizer.transform(X_test['clean_subcategories'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encoding ",X_train_one_hot_clean_sub_cat.shape)
print("Shape of matrix after one hot encoding ",X_cv_one_hot_clean_sub_cat.shape)
print("Shape of matrix after one hot encoding ",X_test_one_hot_clean_sub_cat.shape)

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding (49472, 30)
Shape of matrix after one hot encoding (14372, 30)
Shape of matrix after one hot encoding (21450, 30)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_one_hot_clean_sub_cat.pkl", "wb") as file:
    pickle.dump(X_train_one_hot_clean_sub_cat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_one_hot_clean_sub_cat.pkl", "wb") as file:
    pickle.dump(X_cv_one_hot_clean_sub_cat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_one_hot_clean_sub_cat.pkl", "wb") as file:
    pickle.dump(X_test_one_hot_clean_sub_cat, file)
```

School State

In [0]:

```
# School State
vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_school_state = vectorizer.fit_transform(X_train['school_state'].values)
X_cv_one_hot_clean_school_state = vectorizer.transform(X_cv['school_state'].values)
X_test_one_hot_clean_school_state = vectorizer.transform(X_test['school_state'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encoding ",X_train_one_hot_clean_school_state.shape)
print("Shape of matrix after one hot encoding ",X_cv_one_hot_clean_school_state.shape)
print("Shape of matrix after one hot encoding ",X_test_one_hot_clean_school_state.shape)

['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
Shape of matrix after one hot encoding (49472, 51)
Shape of matrix after one hot encoding (14372, 51)
Shape of matrix after one hot encoding (21450, 51)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_one_hot_clean_school_state.pkl", "wb"):
    pickle.dump(X_train_one_hot_clean_school_state, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_one_hot_clean_school_state.pkl", "wb"):
    pickle.dump(X_cv_one_hot_clean_school_state, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_one_hot_clean_school_state.pkl", "wb"):
    pickle.dump(X_test_one_hot_clean_school_state, file)
```

Teacher Prefix

In [0]:

```
# Teacher Prefix
import re
X_train_prefix_list = []
X_cv_prefix_list = []
X_test_prefix_list = []
for s in tqdm(X_train['teacher_prefix'].values):
    train_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    train_prefix = re.sub('nan', '', str(train_prefix))
    X_train_prefix_list.append(train_prefix)
for s in tqdm(X_cv['teacher_prefix'].values):
    cv_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    X_cv_prefix_list.append(cv_prefix)
for s in tqdm(X_test['teacher_prefix'].values):
    test_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    test_prefix = re.sub('nan', '', str(test_prefix))
    X_test_prefix_list.append(test_prefix)

vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_teacher_prefix = vectorizer.fit_transform(X_train_prefix_list)
X_cv_one_hot_clean_teacher_prefix = vectorizer.transform(X_cv_prefix_list)
X_test_one_hot_clean_teacher_prefix = vectorizer.fit_transform(X_test_prefix_list)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_teacher_prefix.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_teacher_prefix.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_teacher_prefix.shape)
```

```
100%|██████████| 49472/49472 [00:00<00:00, 294272.01it/s]
100%|██████████| 14372/14372 [00:00<00:00, 431725.51it/s]
100%|██████████| 21450/21450 [00:00<00:00, 296679.69it/s]
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig (49472, 5)
Shape of matrix after one hot encodig (14372, 5)
Shape of matrix after one hot encodig (21450, 5)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_one_hot_clean_teacher_pre
pickle.dump(X_train_one_hot_clean_teacher_prefix, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_one_hot_clean_teacher_prefi
pickle.dump(X_cv_one_hot_clean_teacher_prefix, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_one_hot_clean_teacher_pre
pickle.dump(X_test_one_hot_clean_teacher_prefix, file)
```

Project Grade Category

In [0]:

```
# Project Grade Category

def grade_cat_cleaning(data):
    proj_grade_cat_list = []
    for grade in tqdm(data):
        # grade_cat = re.sub('-', ' to ', grade)
        # grade_cat = re.sub('2', ' two ', grade_cat)
        # grade_cat = re.sub('3', ' three ', grade_cat)
        # grade_cat = re.sub('5', ' five ', grade_cat)
        # grade_cat = re.sub('6', ' six ', grade_cat)
        # grade_cat = re.sub('8', ' eight ', grade_cat)
        # grade_cat = re.sub('9', ' nine ', grade_cat)
        # grade_cat = re.sub('12', ' twelve ', grade_cat)
        proj_grade_cat_list.append(grade.lower().strip())
    return proj_grade_cat_list

X_train_proj_grade_cat = grade_cat_cleaning([sent for sent in X_train['project_grade_category']])
X_cv_proj_grade_cat = grade_cat_cleaning([sent for sent in X_cv['project_grade_category']])
X_test_proj_grade_cat = grade_cat_cleaning([sent for sent in X_test['project_grade_category']])
vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_project_grade = vectorizer.fit_transform(X_train_proj_grade_cat)
X_cv_one_hot_clean_project_grade = vectorizer.transform(X_cv_proj_grade_cat)
X_test_one_hot_clean_project_grade = vectorizer.transform(X_test_proj_grade_cat)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encoding ", X_train_one_hot_clean_project_grade.shape)
print("Shape of matrix after one hot encoding ", X_cv_one_hot_clean_project_grade.shape)
print("Shape of matrix after one hot encoding ", X_test_one_hot_clean_project_grade.shape)
```

```
100%|██████████| 49472/49472 [00:00<00:00, 1042617.07it/s]
100%|██████████| 14372/14372 [00:00<00:00, 812012.19it/s]
100%|██████████| 21450/21450 [00:00<00:00, 939395.87it/s]
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encoding (49472, 4)
Shape of matrix after one hot encoding (14372, 4)
Shape of matrix after one hot encoding (21450, 4)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_one_hot_clean_project_grade.pkl", "wb") as file:
    pickle.dump(X_train_one_hot_clean_project_grade, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_one_hot_clean_project_grade.pkl", "wb") as file:
    pickle.dump(X_cv_one_hot_clean_project_grade, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_one_hot_clean_project_grade.pkl", "wb") as file:
    pickle.dump(X_test_one_hot_clean_project_grade, file)
```

Price

In [0]:

```
# check this one: https://www.youtube.com/watch?v=0H0qOcLn3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScaler.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329. ... 399.
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standard deviation
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_price_std = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
X_cv_price_std = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))#
X_test_price_std = price_scalar.transform(X_test['price'].values.reshape(-1, 1))

print("Shape of matrix after one hot encoding ",X_train_price_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_price_std.shape)
print("Shape of matrix after one hot encoding ",X_test_price_std.shape)
```

Mean : 316.4775693321474, Standard deviation : 359.65060192679846

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_price_std.pkl","wb") as file:
    pickle.dump(X_train_price_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_price_std.pkl","wb") as file:
    pickle.dump(X_cv_price_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_price_std.pkl","wb") as file:
    pickle.dump(X_test_price_std, file)
```

teacher_number_of_previously_posted_projects

In [0]:

```
teacher_no_of_projects_scalar = StandardScaler()
teacher_no_of_projects_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values)
print(f"Mean : {teacher_no_of_projects_scalar.mean_[0]}, Standard deviation : {np.sqrt(teacher_no_of_projects_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_teacher_no_of_projects_std = teacher_no_of_projects_scalar.transform(X_train['teacher_number_of_previously_posted_projects'].values)
X_cv_teacher_no_of_projects_std = teacher_no_of_projects_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values)
X_test_teacher_no_of_projects_std = teacher_no_of_projects_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values)

print("Shape of matrix after one hot encoding ",X_train_teacher_no_of_projects_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_teacher_no_of_projects_std.shape)
print("Shape of matrix after one hot encoding ",X_test_teacher_no_of_projects_std.shape)
```

Mean : 9.540002425614489, Standard deviation : 24.382347814122472

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_teacher_no_of_projects_std.pkl", "wb") as file:
    pickle.dump(X_train_teacher_no_of_projects_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_teacher_no_of_projects_std.pkl", "wb") as file:
    pickle.dump(X_cv_teacher_no_of_projects_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_teacher_no_of_projects_std.pkl", "wb") as file:
    pickle.dump(X_test_teacher_no_of_projects_std, file)
```

quantity

In [0]:

```
quantity_scalar = StandardScaler()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard deviation
print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation : {np.sqrt(quantity_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_quantity_std = quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
X_cv_quantity_std = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
X_test_quantity_std = quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))

print("Shape of matrix after one hot encoding ",X_train_quantity_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_quantity_std.shape)
print("Shape of matrix after one hot encoding ",X_test_quantity_std.shape)
```

Mean : 18.822303525226392, Standard deviation : 29.01234391454152

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_quantity_std.pkl","wb")
    pickle.dump(X_train_quantity_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_quantity_std.pkl","wb") as
    pickle.dump(X_cv_quantity_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_quantity_std.pkl","wb") a
    pickle.dump(X_test_quantity_std, file)
```

words_in_project_title

In [0]:

```
words_in_project_title_scalar = StandardScaler()
words_in_project_title_scalar.fit(X_train['words_in_project_title'].values.reshape(-1,1)) #
print(f"Mean : {words_in_project_title_scalar.mean_[0]}, Standard deviation : {np.sqrt(word

# Now standardize the data with above maen and variance.
X_train_words_in_project_title_std = words_in_project_title_scalar.transform(X_train['words
X_cv_words_in_project_title_std = words_in_project_title_scalar.transform(X_cv['words_in_pr
X_test_words_in_project_title_std = words_in_project_title_scalar.transform(X_test['words_i

print("Shape of matrix after one hot encodig ",X_train_words_in_project_title_std.shape)
print("Shape of matrix after one hot encodig ",X_cv_words_in_project_title_std.shape)
print("Shape of matrix after one hot encodig ",X_test_words_in_project_title_std.shape)
```

```
Mean : 28.136865297542045, Standard deviation : 12.03650579710131
Shape of matrix after one hot encodig (49472, 1)
Shape of matrix after one hot encodig (14372, 1)
Shape of matrix after one hot encodig (21450, 1)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_words_in_project_title_s
    pickle.dump(X_train_words_in_project_title_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_words_in_project_title_std.
    pickle.dump(X_cv_words_in_project_title_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_words_in_project_title_st
    pickle.dump(X_test_words_in_project_title_std, file)
```

words_in_essays

In [0]:

```

words_in_essays_scalar = StandardScaler()
words_in_essays_scalar.fit(X_train['words_in_essays'].values.reshape(-1,1)) # finding the mean
print(f"Mean : {words_in_essays_scalar.mean_[0]}, Standard deviation : {np.sqrt(words_in_essays_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_words_in_essays_std = words_in_essays_scalar.transform(X_train['words_in_essays'].values)
X_cv_words_in_essays_std = words_in_essays_scalar.transform(X_cv['words_in_essays'].values)
X_test_words_in_essays_std = words_in_essays_scalar.transform(X_test['words_in_essays'].values)

print("Shape of matrix after one hot encoding ",X_train_words_in_essays_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_words_in_essays_std.shape)
print("Shape of matrix after one hot encoding ",X_test_words_in_essays_std.shape)

```

Mean : 1041.7638664294955, Standard deviation : 278.5106033874048

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_words_in_essays_std.pkl", "wb") as file:
    pickle.dump(X_train_words_in_essays_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_words_in_essays_std.pkl", "wb") as file:
    pickle.dump(X_cv_words_in_essays_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_words_in_essays_std.pkl", "wb") as file:
    pickle.dump(X_test_words_in_essays_std, file)

```

Negative Sentiment: Essay

In [0]:

```

negative_sentiment_scalar = StandardScaler()
negative_sentiment_scalar.fit(X_train['negative'].values.reshape(-1,1)) # finding the mean
print(f"Mean : {negative_sentiment_scalar.mean_[0]}, Standard deviation : {np.sqrt(negative_sentiment_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_negative_sentiment_std = negative_sentiment_scalar.transform(X_train['negative'].values)
X_cv_negative_sentiment_std = negative_sentiment_scalar.transform(X_cv['negative'].values)
X_test_negative_sentiment_std = negative_sentiment_scalar.transform(X_test['negative'].values)

print("Shape of matrix after one hot encoding ",X_train_negative_sentiment_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_negative_sentiment_std.shape)
print("Shape of matrix after one hot encoding ",X_test_negative_sentiment_std.shape)

```

Mean : 0.046292731241914624, Standard deviation : 0.03501994749731875

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_negative_sentiment_std.p
pickle.dump(X_train_negative_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_negative_sentiment_std.pkl"
pickle.dump(X_cv_negative_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_negative_sentiment_std.pk
pickle.dump(X_test_negative_sentiment_std, file)
```

Positive Sentiment: Essay

In [0]:

```
positive_sentiment_scalar = StandardScaler()
positive_sentiment_scalar.fit(X_train['positive'].values.reshape(-1,1)) # finding the mean
print(f"Mean : {positive_sentiment_scalar.mean_[0]}, Standard deviation : {np.sqrt(positive

# Now standardize the data with above mean and variance.
X_train_positive_sentiment_std = positive_sentiment_scalar.transform(X_train['positive'].va
X_cv_positive_sentiment_std = positive_sentiment_scalar.transform(X_cv['positive'].values.r
X_test_positive_sentiment_std = positive_sentiment_scalar.transform(X_test['positive'].valu

print("Shape of matrix after one hot encodig ",X_train_positive_sentiment_std.shape)
print("Shape of matrix after one hot encodig ",X_cv_positive_sentiment_std.shape)
print("Shape of matrix after one hot encodig ",X_test_positive_sentiment_std.shape)
```

```
Mean : 0.269340778622251, Standard deviation : 0.07577294270703104
Shape of matrix after one hot encodig (49472, 1)
Shape of matrix after one hot encodig (14372, 1)
Shape of matrix after one hot encodig (21450, 1)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_positive_sentiment_std.p
pickle.dump(X_train_positive_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_positive_sentiment_std.pkl"
pickle.dump(X_cv_positive_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_positive_sentiment_std.pk
pickle.dump(X_test_positive_sentiment_std, file)
```

Neutral Sentiment: Essay

In [0]:

```
neutral_sentiment_scalar = StandardScaler()
neutral_sentiment_scalar.fit(X_train['neutral'].values.reshape(-1,1)) # finding the mean and variance
print(f"Mean : {neutral_sentiment_scalar.mean_[0]}, Standard deviation : {np.sqrt(neutral_sentiment_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_neutral_sentiment_std = neutral_sentiment_scalar.transform(X_train['neutral'].values.reshape(-1,1))
X_cv_neutral_sentiment_std = neutral_sentiment_scalar.transform(X_cv['neutral'].values.reshape(-1,1))
X_test_neutral_sentiment_std = neutral_sentiment_scalar.transform(X_test['neutral'].values.reshape(-1,1))

print("Shape of matrix after one hot encoding ",X_train_neutral_sentiment_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_neutral_sentiment_std.shape)
print("Shape of matrix after one hot encoding ",X_test_neutral_sentiment_std.shape)
```

Mean : 0.6843688551099613, Standard deviation : 0.0739851262012348

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_neutral_sentiment_std.pkl", "wb") as file:
    pickle.dump(X_train_neutral_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_neutral_sentiment_std.pkl", "wb") as file:
    pickle.dump(X_cv_neutral_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_neutral_sentiment_std.pkl", "wb") as file:
    pickle.dump(X_test_neutral_sentiment_std, file)
```

Compound Sentiment: Essay

In [0]:

```
compound_sentiment_scalar = StandardScaler()
compound_sentiment_scalar.fit(X_train['compound'].values.reshape(-1,1)) # finding the mean and variance
print(f"Mean : {compound_sentiment_scalar.mean_[0]}, Standard deviation : {np.sqrt(compound_sentiment_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_compound_sentiment_std = compound_sentiment_scalar.transform(X_train['compound'].values.reshape(-1,1))
X_cv_compound_sentiment_std = compound_sentiment_scalar.transform(X_cv['compound'].values.reshape(-1,1))
X_test_compound_sentiment_std = compound_sentiment_scalar.transform(X_test['compound'].values.reshape(-1,1))

print("Shape of matrix after one hot encoding ",X_train_compound_sentiment_std.shape)
print("Shape of matrix after one hot encoding ",X_cv_compound_sentiment_std.shape)
print("Shape of matrix after one hot encoding ",X_test_compound_sentiment_std.shape)
```

Mean : 0.9555540689683053, Standard deviation : 0.16424481297108479

Shape of matrix after one hot encoding (49472, 1)

Shape of matrix after one hot encoding (14372, 1)

Shape of matrix after one hot encoding (21450, 1)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_compound_sentiment_std.p
pickle.dump(X_train_compound_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_compound_sentiment_std.pkl"
pickle.dump(X_cv_compound_sentiment_std, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_compound_sentiment_std.pk
pickle.dump(X_test_compound_sentiment_std, file)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [0]:

```
X_train.head(2)
```

Out[93]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcate
57486	p088946	i english second language teacher k 2 students...	listen speak repeat	Literacy_Language	ESL Literacy
13389	p143654	my students come diverse backgrounds they come...	art supplies apah	History_Civics Music_Arts	History_Geogra VisualArts

2.3.1: BOW

Project Title

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train.pkl","rb") as file:
    X_train = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv.pkl","rb") as file:
    X_cv = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test.pkl","rb") as file:
    X_test = pickle.load(file)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/DonorsChoose.csv","rb") as file:
    X_train = pickle.load(file)
```

In [0]:

```
count_vectorizer = CountVectorizer(min_df=10, ngram_range = (1,2), max_features = 5000)#
X_train_bow_project_title = count_vectorizer.fit_transform(X_train['cleaned_project_title'])
X_cv_bow_project_title = count_vectorizer.transform(X_cv['cleaned_project_title'])
X_test_bow_project_title = count_vectorizer.transform(X_test['cleaned_project_title'])
print("Shape of matrix after BOW encoding ",X_train_bow_project_title.shape)
print("Shape of matrix after BOW encoding ",X_cv_bow_project_title.shape)
print("Shape of matrix after BOW encoding ",X_test_bow_project_title.shape)
```

Shape of matrix after BOW encoding (49472, 4869)

Shape of matrix after BOW encoding (14372, 4869)

Shape of matrix after BOW encoding (21450, 4869)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_bow_project_title.pkl","w") as file:
    pickle.dump(X_train_bow_project_title, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_bow_project_title.pkl","wb") as file:
    pickle.dump(X_cv_bow_project_title, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_bow_project_title.pkl","wb") as file:
    pickle.dump(X_test_bow_project_title, file)
```

Essays

In [0]:

```
count_vectorizer = CountVectorizer(min_df=10, ngram_range = (1,2), max_features = 5000)#
X_train_bow_essays = count_vectorizer.fit_transform(X_train['cleaned_essay'])
X_cv_bow_essays = count_vectorizer.transform(X_cv['cleaned_essay'])
X_test_bow_essays = count_vectorizer.transform(X_test['cleaned_essay'])
print("Shape of matrix after BOW encoding ",X_train_bow_essays.shape)
print("Shape of matrix after BOW encoding ",X_cv_bow_essays.shape)
print("Shape of matrix after BOW encoding ",X_test_bow_essays.shape)
```

```
Shape of matrix after BOW encoding (49472, 5000)
Shape of matrix after BOW encoding (14372, 5000)
Shape of matrix after BOW encoding (21450, 5000)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_bow_essays.pkl","wb") as f:
    pickle.dump(X_train_bow_essays, f)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_bow_essays.pkl","wb") as f:
    pickle.dump(X_cv_bow_essays, f)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_bow_essays.pkl","wb") as f:
    pickle.dump(X_test_bow_essays, f)
```

Cleaned Text

In [0]:

```
count_vectorizer_text = CountVectorizer(min_df=10, ngram_range = (1,2), max_features = 5000)
X_train_bow_cleaned_text = count_vectorizer_text.fit_transform(X_train['cleaned_text'])
X_cv_bow_cleaned_text = count_vectorizer_text.transform(X_cv['cleaned_text'])
X_test_bow_cleaned_text = count_vectorizer_text.transform(X_test['cleaned_text'])
print("Shape of matrix after BOW encoding ",X_train_bow_cleaned_text.shape)
print("Shape of matrix after BOW encoding ",X_cv_bow_cleaned_text.shape)
print("Shape of matrix after BOW encoding ",X_test_bow_cleaned_text.shape)
```

```
Shape of matrix after BOW encoding (49472, 5000)
Shape of matrix after BOW encoding (14372, 5000)
Shape of matrix after BOW encoding (21450, 5000)
```

2.3.2: TFIDF Vectorizer

Project Title

In [0]:

```

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer_pt = TfidfVectorizer(min_df=10)
X_train_tfidf_project_title = tfidf_vectorizer_pt.fit_transform(X_train['cleaned_project_title'])
X_cv_tfidf_project_title = tfidf_vectorizer_pt.transform(X_cv['cleaned_project_title'])
X_test_tfidf_project_title = tfidf_vectorizer_pt.transform(X_test['cleaned_project_title'])
print("Shape of matrix after TFIDF encoding ",X_train_tfidf_project_title.shape)
print("Shape of matrix after TFIDF encoding ",X_cv_tfidf_project_title.shape)
print("Shape of matrix after TFIDF encoding ",X_test_tfidf_project_title.shape)

```

Shape of matrix after TFIDF encoding (49472, 2230)

Shape of matrix after TFIDF encoding (14372, 2230)

Shape of matrix after TFIDF encoding (21450, 2230)

In [0]:

```

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_project_title.pkl",
        pickle.dump(X_train_tfidf_project_title, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_project_title.pkl", "w
        pickle.dump(X_cv_tfidf_project_title, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_project_title.pkl",
        pickle.dump(X_test_tfidf_project_title, file)

```

Essays

In [0]:

```

tfidf_vectorizer_essay = TfidfVectorizer(min_df=10, ngram_range = (1,2), max_features = 5000)
X_train_tfidf_essays = tfidf_vectorizer_essay.fit_transform(X_train['cleaned_essay'])
X_cv_tfidf_essays = tfidf_vectorizer_essay.transform(X_cv['cleaned_essay'])
X_test_tfidf_essays = tfidf_vectorizer_essay.transform(X_test['cleaned_essay'])
print("Shape of matrix after TFIDF encoding ",X_train_tfidf_essays.shape)
print("Shape of matrix after TFIDF encoding ",X_cv_tfidf_essays.shape)
print("Shape of matrix after TFIDF encoding ",X_test_tfidf_essays.shape)

```

Shape of matrix after TFIDF encoding (49472, 5000)

Shape of matrix after TFIDF encoding (14372, 5000)

Shape of matrix after TFIDF encoding (21450, 5000)

In [0]:

```

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_essays.pkl", "wb")
        pickle.dump(X_train_tfidf_essays, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_essays.pkl", "wb") as
        pickle.dump(X_cv_tfidf_essays, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_essays.pkl", "wb") as
        pickle.dump(X_test_tfidf_essays, file)

```

Cleaned Text

In [0]:

```
tfidf_vectorizer_cleaned_text = TfidfVectorizer(min_df=10, ngram_range = (1,2), max_features=10000)
X_train_tfidf_cleaned_text = tfidf_vectorizer_cleaned_text.fit_transform(X_train['cleaned_text'])
X_cv_tfidf_cleaned_text = tfidf_vectorizer_cleaned_text.transform(X_cv['cleaned_text'])
X_test_tfidf_cleaned_text = tfidf_vectorizer_cleaned_text.transform(X_test['cleaned_text'])
print("Shape of matrix after TFIDF encoding ",X_train_tfidf_cleaned_text.shape)
print("Shape of matrix after TFIDF encoding ",X_cv_tfidf_cleaned_text.shape)
print("Shape of matrix after TFIDF encoding ",X_test_tfidf_cleaned_text.shape)
```

```
Shape of matrix after TFIDF encoding (49472, 5000)
Shape of matrix after TFIDF encoding (14372, 5000)
Shape of matrix after TFIDF encoding (21450, 5000)
```

2.3.3: Avg W2V

Cleaned Text

In [0]:

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
```

In [0]:

```
X_train_list_of_sent=[]
X_cv_list_of_sent=[]
X_test_list_of_sent=[]
for sent in X_train['cleaned_text'].values:
    X_train_list_of_sent.append(sent.split())
for sent in X_cv['cleaned_text'].values:
    X_cv_list_of_sent.append(sent.split())
for sent in X_test['cleaned_text'].values:
    X_test_list_of_sent.append(sent.split())
```

In [0]:

```
#train data avg w2v
w2v_model=Word2Vec(X_train_list_of_sent,min_count=5,size=50, workers=4)
w2v_words = list(w2v_model.wv.vocab)
```

In [0]:

```
# average Word2Vec
# compute average word2vec for each text.
def avg_w2v(sent_list):
    sent_vectors = []; # the avg-w2v for each sentence is stored in this list
    for sent in tqdm(sent_list): # for each tain sentence
        sent_vec = np.zeros(50) # as word vectors are of zero length
        cnt_words = 0; # num of words with a valid vector in the sentence
        for word in sent: # for each word in a sentence
            if word in w2v_words:
                vec = w2v_model.wv[word]
                sent_vec += vec
                cnt_words += 1
        if cnt_words != 0:
            sent_vec /= cnt_words
        sent_vectors.append(sent_vec)
    print("\n",len(sent_vectors))
    print(len(sent_vectors[0]))
    return sent_vectors
```

In [0]:

```
X_train_avg_w2v_text = avg_w2v([sent.split() for sent in X_train['cleaned_text']])
X_cv_avg_w2v_text = avg_w2v([sent.split() for sent in X_cv['cleaned_text']])
X_test_avg_w2v_text = avg_w2v([sent.split() for sent in X_test['cleaned_text']])
print("Shape of matrix after Avg W2V encodig ",len(X_train_avg_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_cv_avg_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_test_avg_w2v_text))
```

```
100%|██████████| 49472/49472 [04:30<00:00, 182.57it/s]
 0%|          | 0/14372 [00:00<?, ?it/s]
```

```
49472
50
```

```
100%|██████████| 14372/14372 [01:23<00:00, 171.34it/s]
```

```
14372
50
```

```
100%|██████████| 21450/21450 [02:04<00:00, 172.97it/s]
```

```
21450
50
```

```
Shape of matrix after Avg W2V encodig 49472
Shape of matrix after Avg W2V encodig 14372
Shape of matrix after Avg W2V encodig 21450
```


In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_avg_w2v_text.pkl","wb")
    pickle.dump(X_train_avg_w2v_text, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_avg_w2v_text.pkl","wb") as
    pickle.dump(X_cv_avg_w2v_text, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_avg_w2v_text.pkl","wb") a
    pickle.dump(X_test_avg_w2v_text, file)
```

2.3.4: TFIDF Weighted W2V

Cleaned Text

In [0]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_model = TfidfVectorizer()
X_train_tfidf_w2v_model_text = tfidf_model.fit_transform(X_train['cleaned_text'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [0]:

```
# TF-IDF weighted Word2Vec
tfidf_features = tfidf_model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfidf

def tfidf_w2v(sen_list):
    tfidf_w2v_vectors = []; # the tfidf-w2v for each sentence/review is stored in this list
    row=0;

    for sent in tqdm(sen_list): # for each review/sentence
        vector = np.zeros(50) # as word vectors are of zero length
        tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
        for word in sent: # for each word in a review/sentence
            if (word in w2v_words) and (word in tfidf_features):
                vec = w2v_model.wv[word]
                tf_idf = dictionary[word]*(sent.count(word)/len(sent))
                vector += (vec * tf_idf)
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors.append(vector)
        row += 1
    return tfidf_w2v_vectors
```

In [0]:

```
X_train_tfidf_w2v_text = tfidf_w2v([sent.split() for sent in X_train['cleaned_text']])
X_cv_tfidf_w2v_text = tfidf_w2v([sent.split() for sent in X_cv['cleaned_text']])
X_test_tfidf_w2v_text = tfidf_w2v([sent.split() for sent in X_test['cleaned_text']])
print("Shape of matrix after Avg W2V encodig ",len(X_train_tfidf_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_cv_tfidf_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_test_tfidf_w2v_text))
```

```
100%|██████████| 49472/49472 [1:18:29<00:00, 10.67it/s]
100%|██████████| 14372/14372 [23:06<00:00, 10.37it/s]
100%|██████████| 21450/21450 [34:22<00:00, 10.40it/s]
```

```
Shape of matrix after Avg W2V encodig 49472
Shape of matrix after Avg W2V encodig 14372
Shape of matrix after Avg W2V encodig 21450
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_w2v_text.pkl","wb") as file:
    pickle.dump(X_train_tfidf_w2v_text, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_w2v_text.pkl","wb") as file:
    pickle.dump(X_cv_tfidf_w2v_text, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_w2v_text.pkl","wb") as file:
    pickle.dump(X_test_tfidf_w2v_text, file)
```

Merging Categorical and Numerical Features

In [0]:

```
from scipy.sparse import coo_matrix, hstack
print(X_train_one_hot_clean_cat.shape, X_train_one_hot_clean_sub_cat.shape, X_train_one_hot_clean_teacher_prefix.shape, X_train_one_hot_clean_project_title.shape)

X_train_categorical_numerical = hstack([X_train_one_hot_clean_cat, X_train_one_hot_clean_sub_cat, X_train_one_hot_clean_teacher_prefix, X_train_one_hot_clean_project_title,
                                         X_train_price_std, X_train_quantity_std, X_train_words_in_project_title, X_train_negative_sentiment_std, X_train_positive_sentiment_std,
                                         X_train_compound_sentiment_std])

X_cv_categorical_numerical = hstack([X_cv_one_hot_clean_cat, X_cv_one_hot_clean_sub_cat, X_cv_one_hot_clean_teacher_prefix, X_cv_one_hot_clean_project_title,
                                     X_cv_price_std, X_cv_quantity_std, X_cv_words_in_project_title, X_cv_negative_sentiment_std, X_cv_positive_sentiment_std,
                                     X_cv_compound_sentiment_std])

X_test_categorical_numerical = hstack([X_test_one_hot_clean_cat, X_test_one_hot_clean_sub_cat, X_test_one_hot_clean_teacher_prefix, X_test_one_hot_clean_project_title,
                                       X_test_price_std, X_test_quantity_std, X_test_words_in_project_title, X_test_negative_sentiment_std, X_test_positive_sentiment_std,
                                       X_test_compound_sentiment_std])

print(X_train_categorical_numerical.shape, X_cv_categorical_numerical.shape, X_test_categorical_numerical.shape)
```

```
(49472, 9) (49472, 30) (49472, 51) (49472, 5) (49472, 4)
(49472, 108) (14372, 108) (21450, 108)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_categorical_numerical.pkl", "wb") as file:
    pickle.dump(X_train_categorical_numerical, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_categorical_numerical.pkl", "wb") as file:
    pickle.dump(X_cv_categorical_numerical, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_categorical_numerical.pkl", "wb") as file:
    pickle.dump(X_test_categorical_numerical, file)
```

SET 1: Merging: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_categorical_numerical.pkl", "rb") as file:
    X_train_categorical_numerical = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_categorical_numerical.pkl", "rb") as file:
    X_cv_categorical_numerical = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_categorical_numerical.pkl", "rb") as file:
    X_test_categorical_numerical = pickle.load(file)
```

In [0]:

```
X_train_bow_feat = hstack([X_train_bow_project_title, X_train_bow_essays, X_train_categorical_numerical])
X_cv_bow_feat = hstack([X_cv_bow_project_title, X_cv_bow_essays, X_cv_categorical_numerical])
X_test_bow_feat = hstack([X_test_bow_project_title, X_test_bow_essays, X_test_categorical_numerical])
print(X_train_bow_feat.shape)
print(X_cv_bow_feat.shape)
print(X_test_bow_feat.shape)
```

(49472, 9977)

(14372, 9977)

(21450, 9977)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_bow_feat.pkl", "wb") as file:
    pickle.dump(X_train_bow_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_bow_feat.pkl", "wb") as file:
    pickle.dump(X_cv_bow_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_bow_feat.pkl", "wb") as file:
    pickle.dump(X_test_bow_feat, file)
```

SET 2: Merging: categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

In [0]:

```
X_train_tfidf_feat = hstack([X_train_tfidf_project_title, X_train_tfidf_essays, X_train_categorical_numerical])
X_cv_tfidf_feat = hstack([X_cv_tfidf_project_title, X_cv_tfidf_essays, X_cv_categorical_numerical])
X_test_tfidf_feat = hstack([X_test_tfidf_project_title, X_test_tfidf_essays, X_test_categorical_numerical])
print(X_train_tfidf_feat.shape)
print(X_cv_tfidf_feat.shape)
print(X_test_tfidf_feat.shape)
```

(49472, 14755)

(14372, 14755)

(21450, 14755)

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_feat.pkl","wb") as file:
    pickle.dump(X_train_tfidf_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_feat.pkl","wb") as file:
    pickle.dump(X_cv_tfidf_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_feat.pkl","wb") as file:
    pickle.dump(X_test_tfidf_feat, file)
```

Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)

In [0]:

```
X_train_avg_w2v_feat = hstack([X_train_avg_w2v_text, X_train_categorical_numerical])
X_cv_avg_w2v_feat = hstack([X_cv_avg_w2v_text, X_cv_categorical_numerical])
X_test_avg_w2v_feat = hstack([X_test_avg_w2v_text, X_test_categorical_numerical])
print(X_train_avg_w2v_feat.shape)
print(X_cv_avg_w2v_feat.shape)
print(X_test_avg_w2v_feat.shape)
```

```
(49472, 158)
(14372, 158)
(21450, 158)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_avg_w2v_feat.pkl","wb") as file:
    pickle.dump(X_train_avg_w2v_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_avg_w2v_feat.pkl","wb") as file:
    pickle.dump(X_cv_avg_w2v_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_avg_w2v_feat.pkl","wb") as file:
    pickle.dump(X_test_avg_w2v_feat, file)
```

Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

In [0]:

```
X_train_tfidf_w2v_feat = hstack([X_train_tfidf_w2v_text, X_train_categorical_numerical])
X_cv_tfidf_w2v_feat = hstack([X_cv_tfidf_w2v_text, X_cv_categorical_numerical])
X_test_tfidf_w2v_feat = hstack([X_test_tfidf_w2v_text, X_test_categorical_numerical])
print(X_train_tfidf_w2v_feat.shape)
print(X_cv_tfidf_w2v_feat.shape)
print(X_test_tfidf_w2v_feat.shape)
```

```
(49472, 158)
(14372, 158)
(21450, 158)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_w2v_feat.pkl", "wb") as file:
    pickle.dump(X_train_tfidf_w2v_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_w2v_feat.pkl", "wb") as file:
    pickle.dump(X_cv_tfidf_w2v_feat, file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_w2v_feat.pkl", "wb") as file:
    pickle.dump(X_test_tfidf_w2v_feat, file)
```

2.4 Applying Decision Tree on different kind of featurization as mentioned in the instructions

Apply Decision Tree on different kind of featurization as mentioned in the instructions

For Every model that you work on make sure you do the step 2 and step 3 of instructions

2.4.1 Applying Decision Trees on BOW, SET 1

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_bow_feat.pkl", "rb") as file:
    X_train_bow_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_bow_feat.pkl", "rb") as file:
    X_cv_bow_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_bow_feat.pkl", "rb") as file:
    X_test_bow_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_train.pkl", "rb") as file:
    y_train = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_cv.pkl", "rb") as file:
    y_cv = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_test.pkl", "rb") as file:
    y_test = pickle.load(file)
```

In [5]:

```

from scipy.sparse import coo_matrix, hstack, vstack
print(X_train_bow_feat.shape)
print(X_cv_bow_feat.shape)
X_train_new = vstack((X_train_bow_feat, X_cv_bow_feat))
y_train_new = pd.concat([y_train, y_cv], axis = 0)
print(X_train_new.shape)
# print(y_train.shape)
# print(y_cv.shape)
print(y_train_new.shape)

```

```

(49472, 14755)
(14372, 14755)
(63844, 14755)
(63844,)

```

In [0]:

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score

```

In [0]:

```

import sklearn
print(sklearn.__version__)

```

0.21.2

In [0]:

```

param_grid = {'max_depth': [1, 5, 10, 50, 100, 500, 1000, 2000], 'min_samples_split' : [5,

```

In [0]:

```

grid_search_dt_bow = GridSearchCV(DecisionTreeClassifier(class_weight = "balanced"), param_
grid_search_dt_bow.fit(X_train_new,y_train_new)
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_bow.pkl","wb") a
    pickle.dump(grid_search_dt_bow, file)

```

In [0]:

```

with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_bow.pkl","rb") a
    grid_search_dt_bow = pickle.load(file)

```

In [14]:

```

print(grid_search_dt_bow.best_params_)

```

```

{'max_depth': 500, 'min_samples_split': 10}

```

In [20]:

```
#https://qiita.com/bmj0114/items/8009f282c99b77780563
test_scores = pd.DataFrame(grid_search_dt_bow.cv_results_['mean_test_score'].reshape(len(p
test_scores.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                      index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                      inplace = True)
test_scores
```

Out[20]:

	5	10	100	500
1	0.566608	0.566608	0.566608	0.566608
5	0.672341	0.672323	0.671606	0.671176
10	0.760482	0.760553	0.746543	0.727856
50	0.870490	0.869119	0.847932	0.796724
100	0.869933	0.870846	0.852762	0.804930
500	0.873452	0.874383	0.855879	0.807175
1000	0.874114	0.874352	0.856879	0.807022
2000	0.873874	0.872117	0.856551	0.806887

In [24]:

```
#https://qiita.com/bmj0114/items/8009f282c99b77780563
train_scores = pd.DataFrame(grid_search_dt_bow.cv_results_['mean_train_score'].reshape(len(
train_scores.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                      index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                      inplace = True)
train_scores
```

Out[24]:

	5	10	100	500
1	0.568542	0.568542	0.568542	0.568542
5	0.681950	0.681931	0.681290	0.679800
10	0.804925	0.804590	0.782406	0.753983
50	0.991178	0.988958	0.945373	0.870147
100	0.998988	0.998409	0.966739	0.895293
500	0.999959	0.999604	0.971911	0.900914
1000	0.999957	0.999608	0.971598	0.901313
2000	0.999959	0.999610	0.972301	0.901901

In [41]:

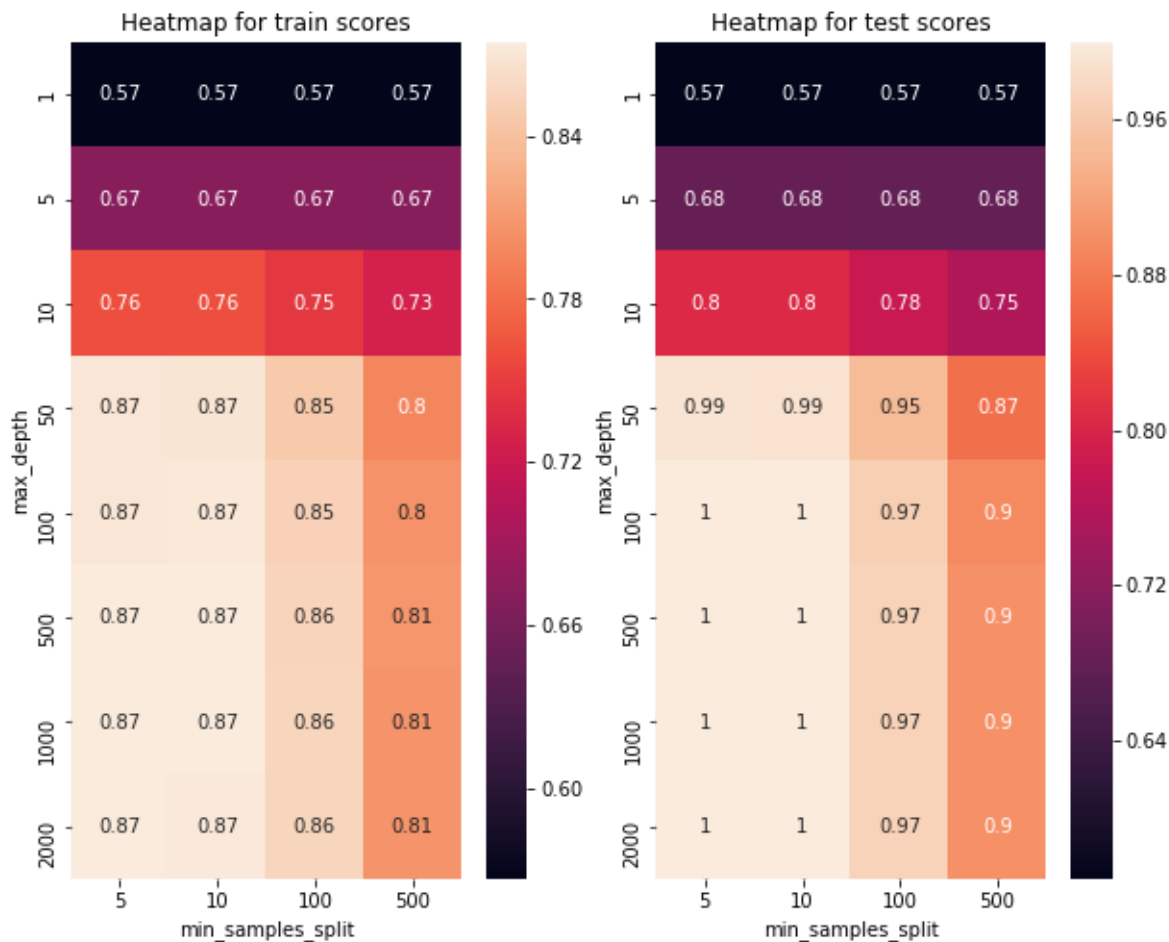
```
train_scores.subtract(test_scores)
```

Out[41]:

	5	10	100	500
1	0.001934	0.001934	0.001934	0.001934
5	0.009609	0.009609	0.009684	0.008624
10	0.044444	0.044037	0.035863	0.026127
50	0.120688	0.119839	0.097441	0.073423
100	0.129055	0.127563	0.113977	0.090364
500	0.126507	0.125222	0.116032	0.093738
1000	0.125844	0.125256	0.114719	0.094291
2000	0.126085	0.127492	0.115751	0.095014

In [40]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 8))
plt.xlabel('max_depth')
sns.heatmap(train_scores, annot=True, xticklabels=param_grid['min_samples_split'], yticklabels=param_grid['max_depth'])
sns.heatmap(test_scores, annot=True, xticklabels=param_grid['min_samples_split'], yticklabels=param_grid['max_depth'])
ax[0].set_title("Heatmap for train scores")
ax[0].set_xlabel = 'min_samples_split', ylabel = 'max_depth'
ax[1].set_title("Heatmap for test scores")
ax[1].set_xlabel = 'min_samples_split', ylabel = 'max_depth'
fig.show()
```



In [58]:

```
clf_bow_1 = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 50, min_samples_split=500)
clf_bow_1.fit(X_train_new, y_train_new)
```

Out[58]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=50,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

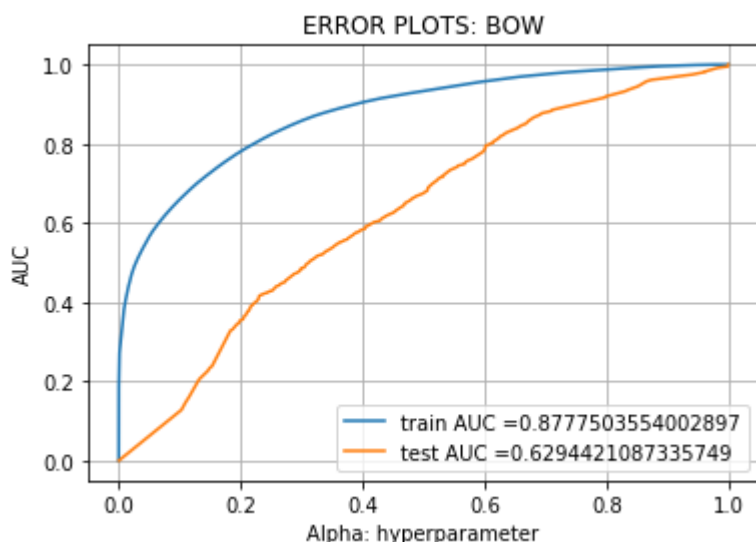
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred_test = clf_bow_1.predict_proba(X_train_new)[: ,1]
y_test_pred_test = clf_bow_1.predict_proba(X_test_bow_feat)[: ,1]

train_fpr_test, train_tpr_test, tr_thresholds_test = roc_curve(y_train_new, y_train_pred_test)
test_fpr_test, test_tpr_test, te_thresholds_test = roc_curve(y_test, y_test_pred_test)
```

In [60]:

```
plt.plot(train_fpr_test, train_tpr_test, label="train AUC =" + str(auc(train_fpr_test, train_tpr_test)))
plt.plot(test_fpr_test, test_tpr_test, label="test AUC =" + str(auc(test_fpr_test, test_tpr_test)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [51]:

```
clf_bow = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 10, min_samples_split=500)
clf_bow.fit(X_train_new,y_train_new)
```

Out[51]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=500,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')
```

In [0]:

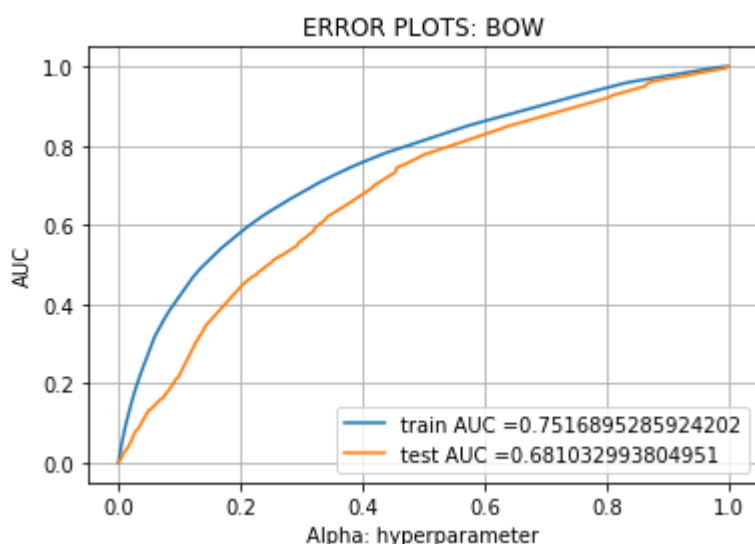
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred = clf_bow.predict_proba(X_train_new)[: ,1]
y_test_pred = clf_bow.predict_proba(X_test_bow_feat)[: ,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_new, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [54]:

```
plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [56]:

```
print(type(predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
print(type(y_test))
```

```
the maximum value of tpr*(1-fpr) 0.4082764138021635 for threshold 0.487
<class 'list'>
<class 'pandas.core.series.Series'>
```

Confusion Matrix for max_depth = 10 and min_samples_split = 500

In [62]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix_train = confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
print(confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
conf_matrix_test = confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.47839122879972085 for threshold 0.483
the maximum value of tpr*(1-fpr) 0.47839122879972085 for threshold 0.483
[[19319  7605]
 [12305 24615]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4082764138021635 for threshold 0.487
the maximum value of tpr*(1-fpr) 0.4082764138021635 for threshold 0.487
[[ 2142  1123]
 [ 6868 11317]]
```

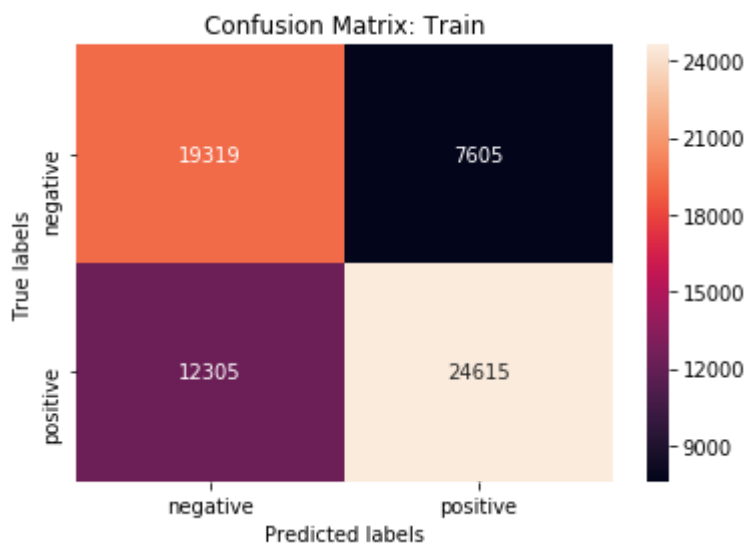
In [69]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix_train, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Train')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[69]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



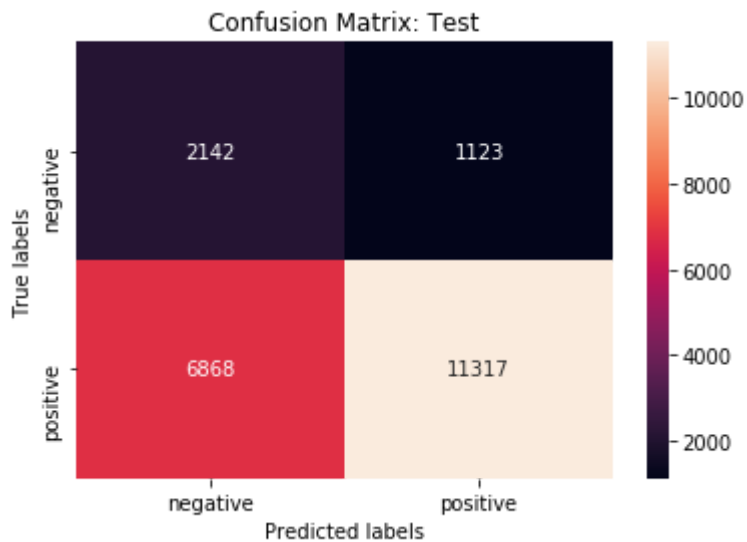
In [70]:

```
heat_map = plt.subplot()
sn.heatmap(conf_matrix_test, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Test')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[70]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



Confusion Matrix for max_depth = 50 and min_samples_split = 500

In [66]:

```

print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix_train_1 = confusion_matrix(y_train_new, predict(y_train_pred_test, tr_thresholds_test, train_fpr_test))
print(confusion_matrix(y_train_new, predict(y_train_pred_test, tr_thresholds_test, train_fpr_test)))
print("Test confusion matrix")
conf_matrix_test_1 = confusion_matrix(y_test, predict(y_test_pred_test, tr_thresholds_test, test_fpr_test))
print(confusion_matrix(y_test, predict(y_test_pred_test, tr_thresholds_test, test_fpr_test)))

```

```

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.6247269932636514 for threshold 0.493
the maximum value of tpr*(1-fpr) 0.6247269932636514 for threshold 0.493
[[21561  5363]
 [ 8118 28802]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.35191501670974806 for threshold 0.643
the maximum value of tpr*(1-fpr) 0.35191501670974806 for threshold 0.643
[[ 1997  1268]
 [ 7722 10463]]

```

In [71]:

```

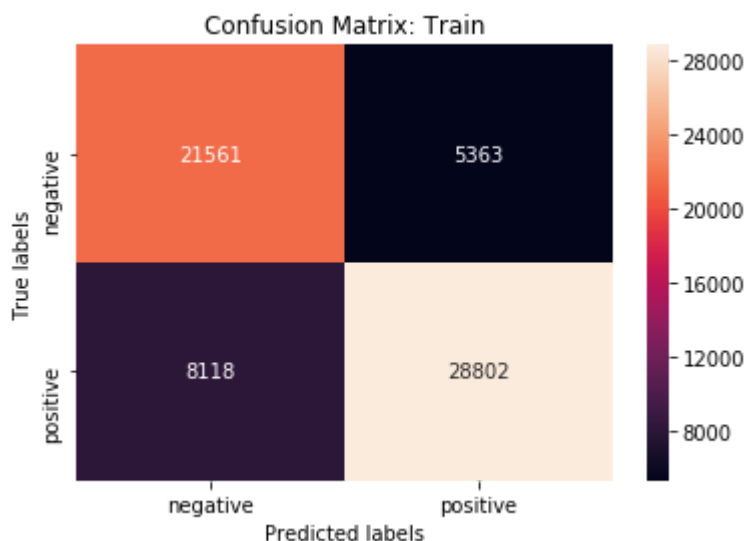
heat_map = plt.subplot()
sn.heatmap(conf_matrix_train_1, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Train')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])

```

Out[71]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



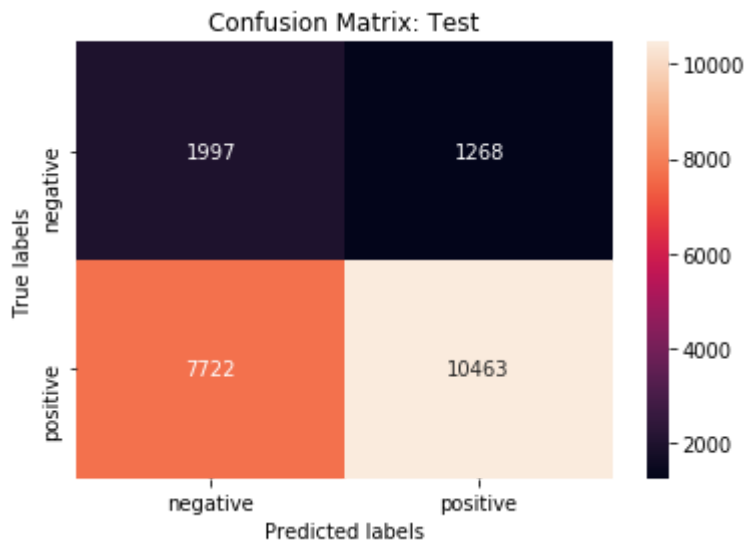
In [72]:

```
heat_map = plt.subplot()
sn.heatmap(conf_matrix_test_1, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Test')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[72]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test.pkl","rb") as file:
    X_test = pickle.load(file)
```

In [75]:

```
print(X_test.shape)
y_test_new = list(y_test)
X_test_new = X_test.copy()
X_test_new['y_actual'] = y_test_new
X_test_new['y_predicted'] = predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)
print(X_test_new.shape)
```

```
(21450, 18)
the maximum value of tpr*(1-fpr) 0.4082764138021635 for threshold 0.487
(21450, 20)
```

In [76]:

```
fp_points = X_test_new[(X_test_new['y_actual'] == 0) & (X_test_new['y_predicted'] == 1)]  
fp_points.shape
```

Out[76]:

(1123, 20)

In [0]:

```
from wordcloud import WordCloud, STOPWORDS  
import matplotlib.pyplot as plt  
import pandas as pd
```

Word Cloud: Essay

In [78]:

```
comment_words = ' '
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in fp_points.cleaned_essay:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

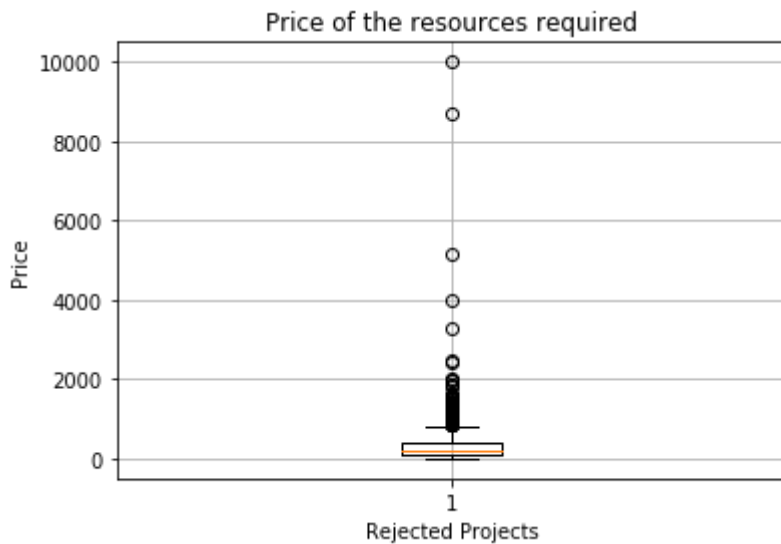
wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```


In [79]:

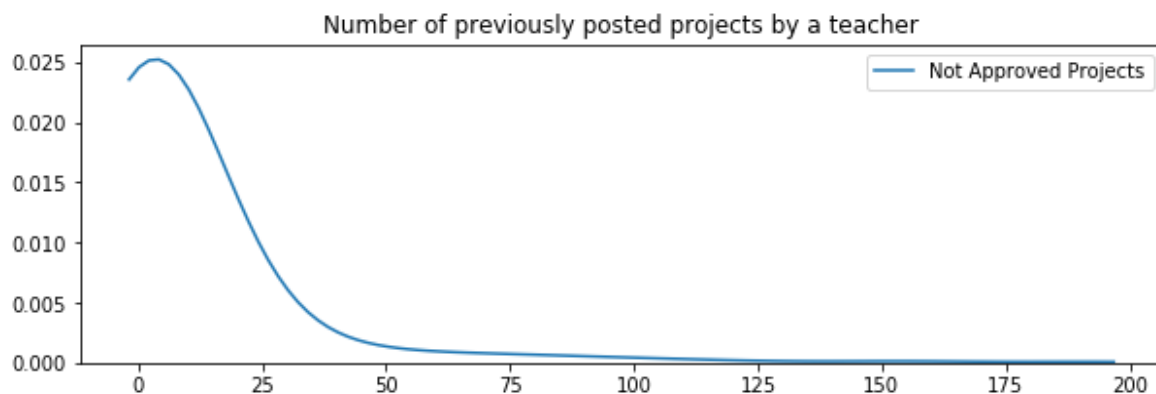
```
plt.boxplot(fp_points['price'])#[fp_points[fp_points['y_actual']==1]['price'], fp_points[fp
# plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.xlabel('Rejected Projects')
plt.ylabel('Price')
plt.title('Price of the resources required')
plt.grid()
plt.show()
```



PDF: teacher_number_of_previously_posted_projects

In [80]:

```
plt.figure(figsize=(10,3))
# sns.kdeplot(fp_points[fp_points['y_actual']==1]['teacher_number_of_previously_posted_proj
sns.kdeplot(fp_points['teacher_number_of_previously_posted_projects'],label="Not Approved P
plt.title('Number of previously posted projects by a teacher')
plt.legend()
plt.show()
```



In [0]:

```
fp_points.columns
```

Out[42]:

```
Index(['id', 'cleaned_essay', 'cleaned_project_title', 'clean_categories',
      'clean_subcategories', 'teacher_prefix', 'school_state',
      'project_grade_category',
      'teacher_number_of_previously_posted_projects', 'cleaned_text', 'price',
      'quantity', 'words_in_project_title', 'words_in_essays', 'negative',
      'positive', 'neutral', 'compound', 'y_actual', 'y_predicted'],
      dtype='object')
```

2.4.1.1 Graphviz visualization of Decision Tree on BOW, SET 1

In [0]:

```
X_train_bow_graph = vstack([X_train_bow_cleaned_text, X_cv_bow_cleaned_text])
```

In [0]:

```
dt_clf_bow = DecisionTreeClassifier(max_depth = 3)
dt_clf_bow.fit(X_train_bow_graph, y_train_new)
```

Out[54]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [0]:

```
import graphviz
from sklearn import tree
dot_data_bow = tree.export_graphviz(dt_clf_bow, out_file=None, feature_names=count_vectorizer.get_feature_names())
```

In [0]:

```
graph_bow = graphviz.Source(dot_data_bow)
graph_bow
```

Out[56]:

```
<graphviz.files.Source at 0x7f2ef13a5ac8>
```

2.4.2 Applying Decision Trees on TFIDF, SET 2

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_feat.pkl","rb") as f:
    X_train_tfidf_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_feat.pkl","rb") as f:
    X_cv_tfidf_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_feat.pkl","rb") as f:
    X_test_tfidf_feat = pickle.load(file)
```

In [82]:

```
X_train_new_tfidf = vstack([X_train_tfidf_feat, X_cv_tfidf_feat])
y_train_new = y_train.append(y_cv)
print(X_train_new_tfidf.shape)
print(y_train_new.shape)
```

```
(63844, 14755)
(63844,)
```

In [0]:

```
grid_search_dt_tfidf = GridSearchCV(DecisionTreeClassifier(class_weight = "balanced"), param_grid=param_grid, cv=5)
grid_search_dt_tfidf.fit(X_train_new_tfidf,y_train_new)
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_tfidf.pkl","wb") as f:
    pickle.dump(grid_search_dt_tfidf, f)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_tfidf.pkl","rb") as f:
    grid_search_dt_tfidf = pickle.load(f)
```

In [84]:

```
print(grid_search_dt_tfidf.best_params_)
```

```
{'max_depth': 500, 'min_samples_split': 5}
```

In [85]:

```
print(grid_search_dt_tfidf.best_estimator_)
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
500,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=5,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [86]:

```
test_scores_tfidf = pd.DataFrame(grid_search_dt_tfidf.cv_results_['mean_test_score'].reshape(
test_scores_tfidf.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                        index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                        inplace = True)
test_scores_tfidf
```

Out[86]:

	5	10	100	500
1	0.566608	0.566608	0.566608	0.566608
5	0.672660	0.672714	0.671940	0.671706
10	0.755721	0.755747	0.742397	0.726808
50	0.863262	0.861548	0.848073	0.811263
100	0.867902	0.866368	0.855182	0.819420
500	0.872747	0.870311	0.859892	0.822138
1000	0.871676	0.870998	0.858894	0.821817
2000	0.870966	0.872618	0.859679	0.821730

In [87]:

```
train_scores_tfidf = pd.DataFrame(grid_search_dt_tfidf.cv_results_['mean_train_score'].resha
train_scores_tfidf.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                        index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                        inplace = True)
train_scores_tfidf
```

Out[87]:

	5	10	100	500
1	0.568542	0.568542	0.568542	0.568542
5	0.684891	0.684891	0.683817	0.683272
10	0.808225	0.807894	0.786773	0.760272
50	0.991705	0.989993	0.953605	0.889172
100	0.998798	0.998617	0.972550	0.914592
500	0.999977	0.999769	0.977416	0.922859
1000	0.999974	0.999772	0.977840	0.922728
2000	0.999973	0.999774	0.977870	0.921752

In [88]:

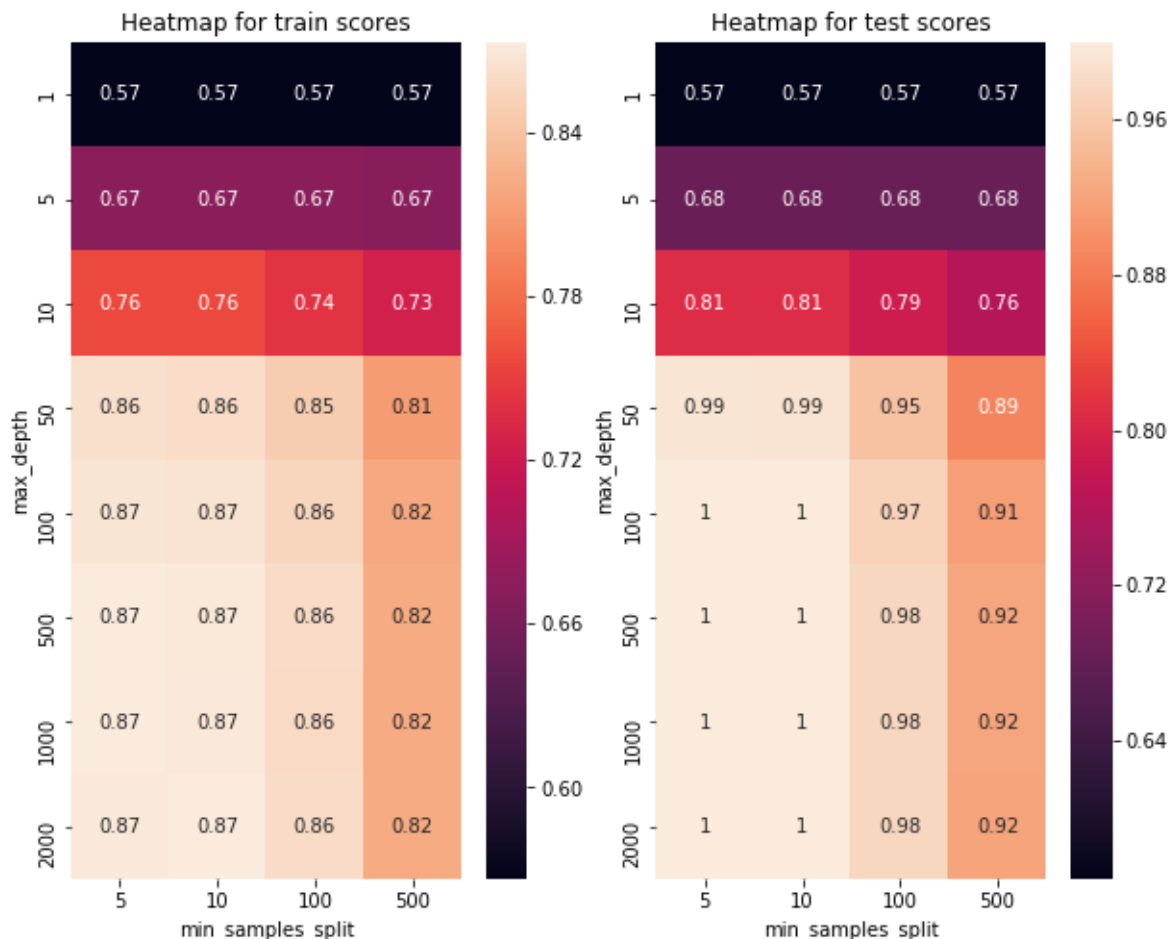
```
train_scores_tfidf.subtract(test_scores_tfidf)
```

Out[88]:

	5	10	100	500
1	0.001934	0.001934	0.001934	0.001934
5	0.012231	0.012177	0.011877	0.011566
10	0.052505	0.052147	0.044377	0.033464
50	0.128443	0.128445	0.105532	0.077909
100	0.130896	0.132249	0.117368	0.095172
500	0.127229	0.129458	0.117524	0.100721
1000	0.128299	0.128774	0.118946	0.100910
2000	0.129006	0.127156	0.118191	0.100022

In [90]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 8))
plt.xlabel('max_depth')
sns.heatmap(train_scores_tfidf, annot=True, xticklabels=param_grid['min_samples_split'], yticklabels=param_grid['max_depth'], yti
sns.heatmap(test_scores_tfidf, annot=True, xticklabels=param_grid['min_samples_split'], yticklabels=param_grid['max_depth'], yti
ax[0].set_title("Heatmap for train scores")
ax[0].set(xlabel = 'min_samples_split', ylabel = 'max_depth')
ax[1].set_title("Heatmap for test scores")
ax[1].set(xlabel = 'min_samples_split', ylabel = 'max_depth')
fig.show()
```



In [95]:

```
clf_tfidf_1 = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 50, min_samples
clf_tfidf_1.fit(X_train_new_tfidf,y_train_new)
```

Out[95]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
50,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

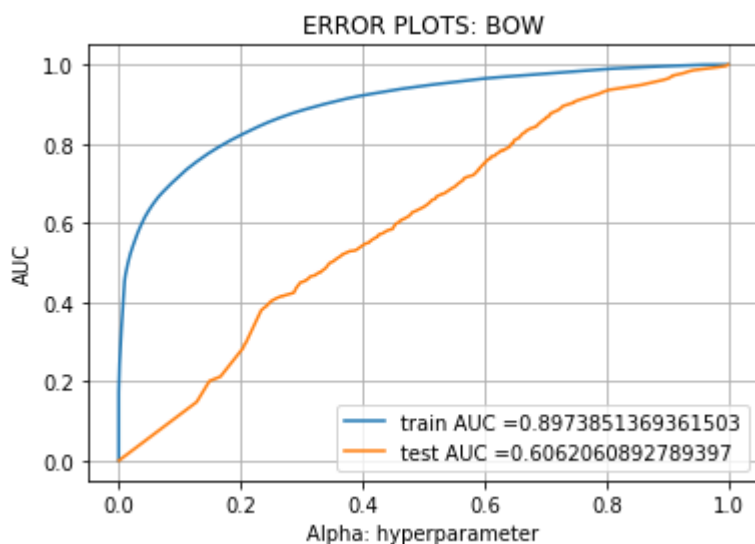
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred_test = clf_tfidf_1.predict_proba(X_train_new_tfidf)[: ,1]
y_test_pred_test = clf_tfidf_1.predict_proba(X_test_tfidf_feat)[: ,1]

train_fpr_test, train_tpr_test, tr_thresholds_test = roc_curve(y_train_new, y_train_pred_test)
test_fpr_test, test_tpr_test, te_thresholds_test = roc_curve(y_test, y_test_pred_test)
```

In [97]:

```
plt.plot(train_fpr_test, train_tpr_test, label="train AUC =" + str(auc(train_fpr_test, train_tpr_test)))
plt.plot(test_fpr_test, test_tpr_test, label="test AUC =" + str(auc(test_fpr_test, test_tpr_test)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [98]:

```
clf_tfidf = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 10, min_samples_split=10)
clf_tfidf.fit(X_train_new_tfidf,y_train_new)
```

Out[98]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

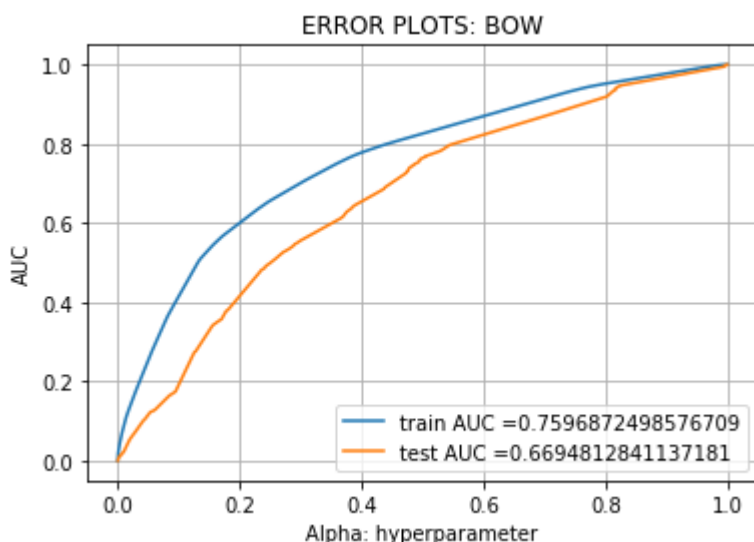
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred = clf_tfidf.predict_proba(X_train_new_tfidf)[: ,1]
y_test_pred = clf_tfidf.predict_proba(X_test_tfidf_feat)[: ,1]

train_fpr, train_tpr, tr_thresholds= roc_curve(y_train_new, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [157]:

```
plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [105]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix_train = confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
print(confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
conf_matrix_test = confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4923798286674437 for threshold 0.476
the maximum value of tpr*(1-fpr) 0.4923798286674437 for threshold 0.476
[[20005  6919]
 [12454 24466]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.39415508717827363 for threshold 0.476
the maximum value of tpr*(1-fpr) 0.39415508717827363 for threshold 0.476
[[ 1998  1267]
 [ 6472 11713]]
```

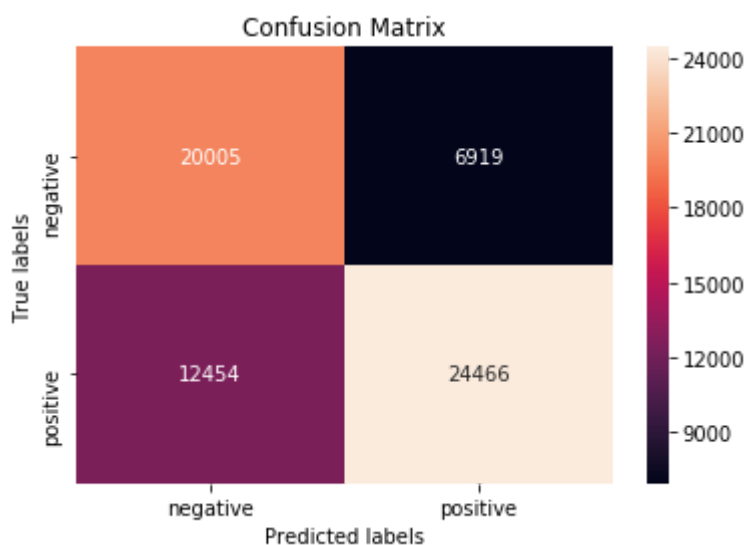
In [106]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix_train, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Train')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[106]:

[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]



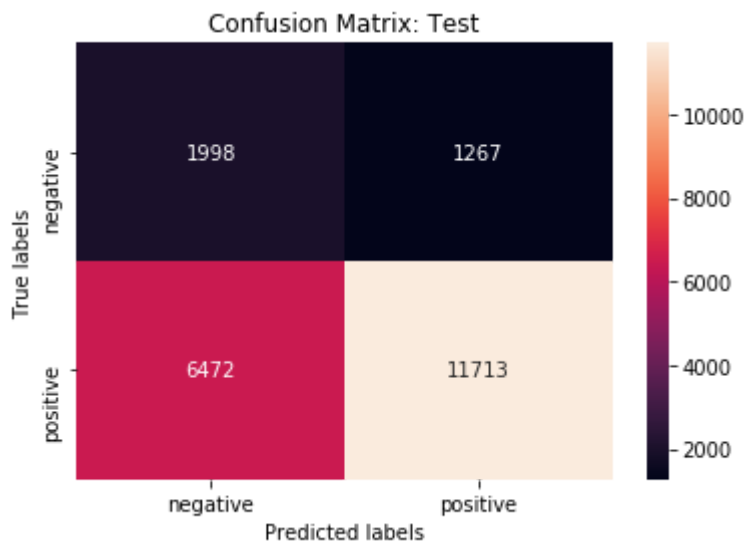
In [107]:

```
heat_map = plt.subplot()
sn.heatmap(conf_matrix_test, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Test')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[107]:

[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]



In [158]:

```
print(X_test.shape)
y_test_new = list(y_test)
X_test_new_tfidf = X_test.copy()
X_test_new_tfidf['y_actual'] = y_test_new
X_test_new_tfidf['y_predicted'] = predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)
print(X_test_new_tfidf.shape)
```

(21450, 18)
the maximum value of $tpr \cdot (1 - fpr)$ 0.39415508717827363 for threshold 0.476
(21450, 20)

In [159]:

```
fp_points_tfidf = X_test_new_tfidf[(X_test_new_tfidf['y_actual'] == 0) & (X_test_new_tfidf['y_predicted'] == 1)]
fp_points_tfidf.shape
```

Out[159]:

(1267, 20)

Word Cloud: Essay

In [160]:

```
comment_words = ' '
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in fp_points_tfidf.cleaned_essay:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

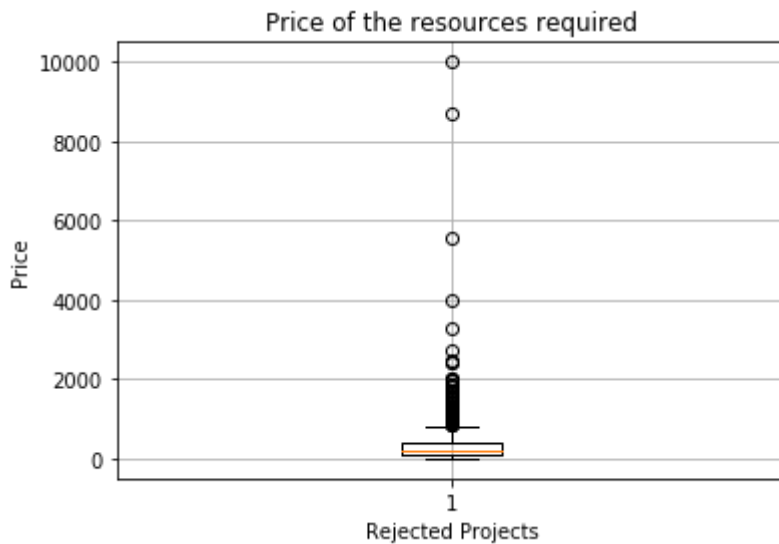
wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```


In [161]:

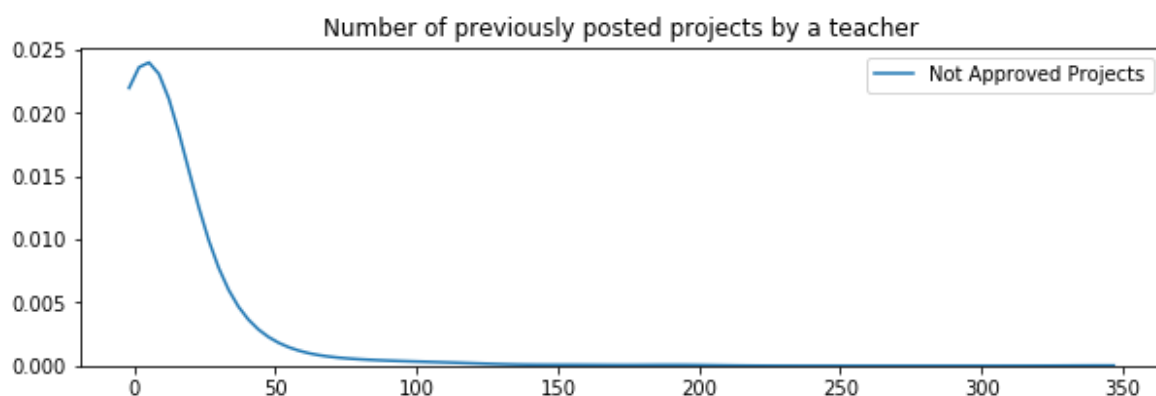
```
plt.boxplot(fp_points_tfidf['price'])#[fp_points[fp_points['y_actual']==1]['price'], fp_poi
# plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.xlabel('Rejected Projects')
plt.ylabel('Price')
plt.title('Price of the resources required')
plt.grid()
plt.show()
```



PDF: teacher_number_of_previously_posted_projects

In [162]:

```
plt.figure(figsize=(10,3))
# sns.kdeplot(fp_points[fp_points['y_actual']==1]['teacher_number_of_previously_posted_proj
sns.kdeplot(fp_points_tfidf['teacher_number_of_previously_posted_projects'],label="Not Appr
plt.title('Number of previously posted projects by a teacher')
plt.legend()
plt.show()
```



2.4.2.1 Graphviz visualization of Decision Tree on TFIDF, SET 2

In [0]:

```
X_train_tfidf_graph = vstack([X_train_tfidf_cleaned_text, X_cv_tfidf_cleaned_text])
```

In [0]:

```
dt_clf_tfidf = DecisionTreeClassifier(max_depth = 3)
dt_clf_tfidf.fit(X_train_tfidf_graph,y_train_new)
```

Out[95]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

```
import graphviz
from sklearn import tree
dot_data_tfidf = tree.export_graphviz(dt_clf_tfidf, out_file=None, feature_names=tfidf_vect
```

In [0]:

```
graph_tfidf = graphviz.Source(dot_data_tfidf)
graph_tfidf
```

Out[97]:

```
<graphviz.files.Source at 0x7f2ef69a3e10>
```

2.4.3 Applying Decision Trees on AVG W2V, SET 3

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_avg_w2v_feat.pkl","rb")
    X_train_avg_w2v_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_avg_w2v_feat.pkl","rb") as
    X_cv_avg_w2v_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_avg_w2v_feat.pkl","rb") as
    X_test_avg_w2v_feat = pickle.load(file)
```

In [109]:

```
X_train_new_avg_w2v = vstack([X_train_avg_w2v_feat, X_cv_avg_w2v_feat])
y_train_new = y_train.append(y_cv)
print(X_train_new_avg_w2v.shape)
print(y_train_new.shape)
```

```
(63844, 158)
(63844,)
```

In [0]:

```
grid_search_dt_avg_w2v = GridSearchCV(DecisionTreeClassifier(class_weight = "balanced"), pa
grid_search_dt_avg_w2v.fit(X_train_new_avg_w2v,y_train_new)
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_avg_w2v.pkl","wb
pickle.dump(grid_search_dt_avg_w2v, file)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_avg_w2v.pkl","rb
grid_search_dt_avg_w2v = pickle.load(file)
```

In [111]:

```
print(grid_search_dt_avg_w2v.best_params_)
```

```
{'max_depth': 50, 'min_samples_split': 10}
```

In [112]:

```
print(grid_search_dt_avg_w2v.best_estimator_)
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
50,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=10,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [113]:

```
test_scores_avg_w2v = pd.DataFrame(grid_search_dt_avg_w2v.cv_results_['mean_test_score'].re
test_scores_avg_w2v.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                           index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                           inplace = True)
test_scores_avg_w2v
```

Out[113]:

	5	10	100	500
1	0.566608	0.566608	0.566608	0.566608
5	0.678729	0.678729	0.678567	0.677597
10	0.782265	0.781409	0.761924	0.729017
50	0.874333	0.877383	0.839309	0.743883
100	0.874097	0.877005	0.838486	0.743695
500	0.874845	0.877076	0.839753	0.743804
1000	0.874216	0.877097	0.839193	0.743704
2000	0.875311	0.877243	0.838321	0.743873

In [114]:

```
train_scores_avg_w2v = pd.DataFrame(grid_search_dt_avg_w2v.cv_results_['mean_train_score'].
train_scores_avg_w2v.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                             index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                             inplace = True)
train_scores_avg_w2v
```

Out[114]:

	5	10	100	500
1	0.568542	0.568542	0.568542	0.568542
5	0.692266	0.692266	0.691903	0.690322
10	0.846757	0.845897	0.814937	0.765635
50	0.999900	0.999207	0.937113	0.793081
100	0.999903	0.999195	0.936880	0.793269
500	0.999902	0.999204	0.936886	0.793118
1000	0.999906	0.999208	0.937008	0.793252
2000	0.999904	0.999205	0.937286	0.793241

In [115]:

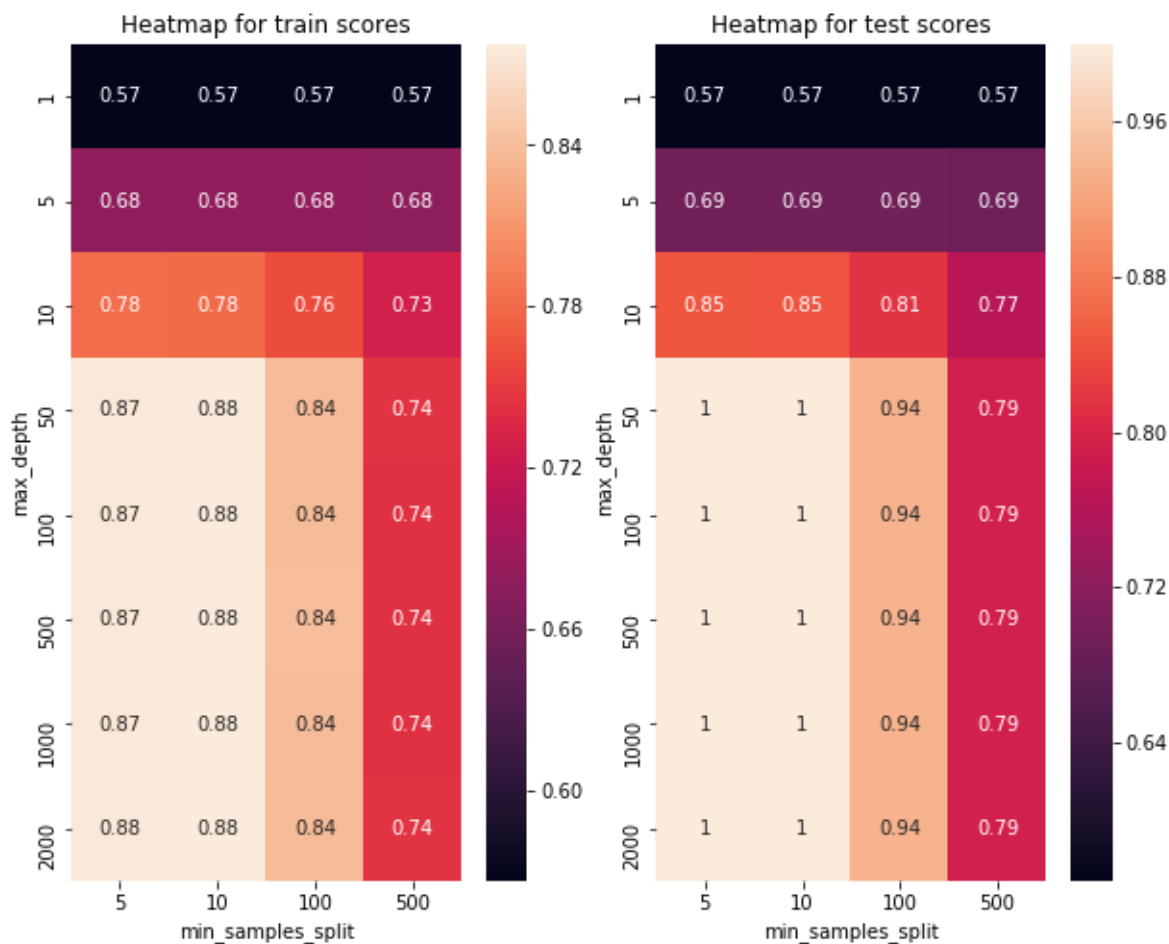
```
train_scores_avg_w2v.subtract(test_scores_avg_w2v)
```

Out[115]:

	5	10	100	500
1	0.001934	0.001934	0.001934	0.001934
5	0.013537	0.013537	0.013336	0.012726
10	0.064492	0.064487	0.053012	0.036618
50	0.125566	0.121824	0.097804	0.049198
100	0.125806	0.122191	0.098394	0.049575
500	0.125057	0.122128	0.097134	0.049314
1000	0.125691	0.122111	0.097815	0.049548
2000	0.124593	0.121963	0.098965	0.049368

In [116]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 8))
plt.xlabel('max_depth')
sns.heatmap(train_scores_avg_w2v, annot=True, xticklabels=param_grid['min_samples_split'],
sns.heatmap(test_scores_avg_w2v, annot=True, xticklabels=param_grid['min_samples_split'], y
ax[0].set_title("Heatmap for train scores")
ax[0].set(xlabel = 'min_samples_split', ylabel = 'max_depth')
ax[1].set_title("Heatmap for test scores")
ax[1].set(xlabel = 'min_samples_split', ylabel = 'max_depth')
fig.show()
```



In [117]:

```
clf_avg_w2v_1 = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 50, min_samp
clf_avg_w2v_1.fit(X_train_new_avg_w2v,y_train_new)
```

Out[117]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
50,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

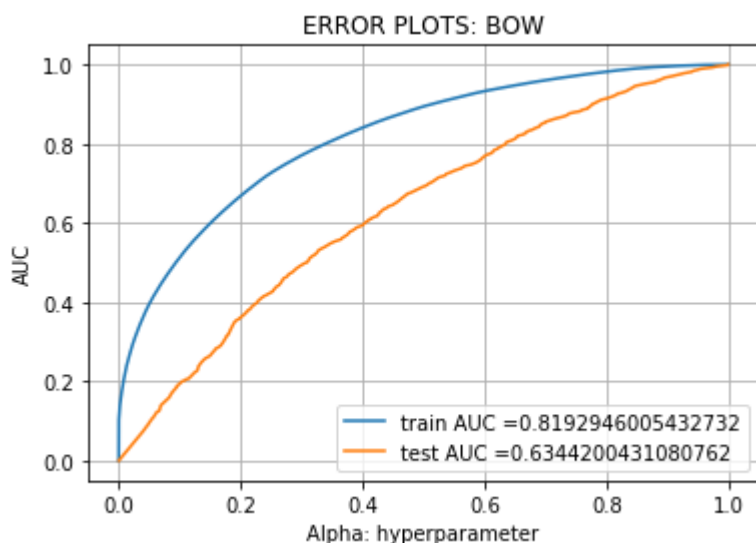
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred_test = clf_avg_w2v_1.predict_proba(X_train_new_avg_w2v)[: ,1]
y_test_pred_test = clf_avg_w2v_1.predict_proba(X_test_avg_w2v_feat)[: ,1]

train_fpr_test, train_tpr_test, tr_thresholds_test = roc_curve(y_train_new, y_train_pred_te
test_fpr_test, test_tpr_test, te_thresholds_test = roc_curve(y_test, y_test_pred_test)
```

In [120]:

```
plt.plot(train_fpr_test, train_tpr_test, label="train AUC =" +str(auc(train_fpr_test, train_
plt.plot(test_fpr_test, test_tpr_test, label="test AUC =" +str(auc(test_fpr_test, test_tpr_t
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [121]:

```
clf_avd_w2v = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 10, min_samples
clf_avd_w2v.fit(X_train_new_avg_w2v,y_train_new)
```

Out[121]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
10,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

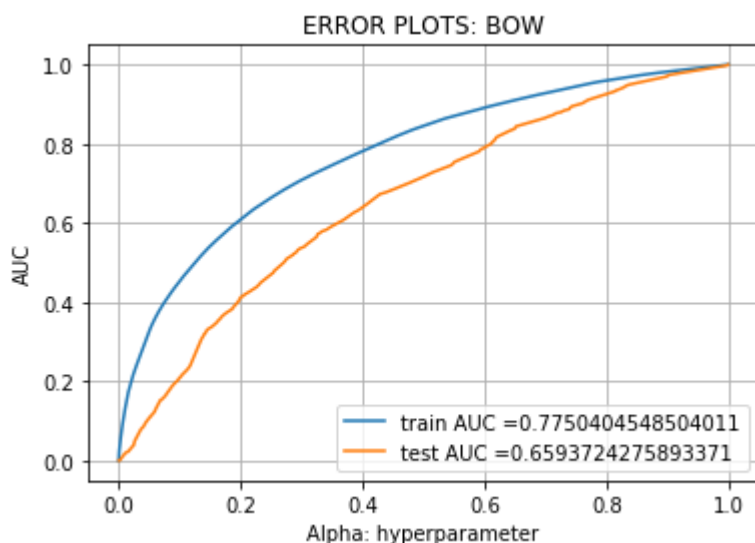
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred = clf_avd_w2v.predict_proba(X_train_new_avg_w2v)[: ,1]
y_test_pred = clf_avd_w2v.predict_proba(X_test_avg_w2v_feat)[: ,1]

train_fpr, train_tpr, tr_thresholds= roc_curve(y_train_new, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [146]:

```
plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```

# we are writing our own function for predict, with defined threshould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t, 2))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions

```

In [148]:

```

print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix_train = confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
print(confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
conf_matrix_test = confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4986533952638726 for threshold 0.486
the maximum value of tpr*(1-fpr) 0.4986533952638726 for threshold 0.486
[[19669  7255]
 [11719 25201]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.38601213240975324 for threshold 0.553
the maximum value of tpr*(1-fpr) 0.38601213240975324 for threshold 0.553
[[ 2178  1087]
 [ 7662 10523]]

```

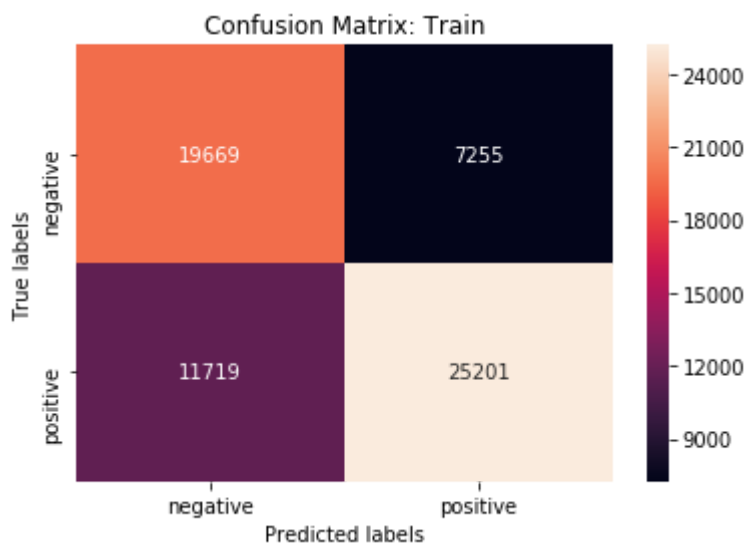
In [149]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix_train, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Train')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[149]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



In [150]:

```

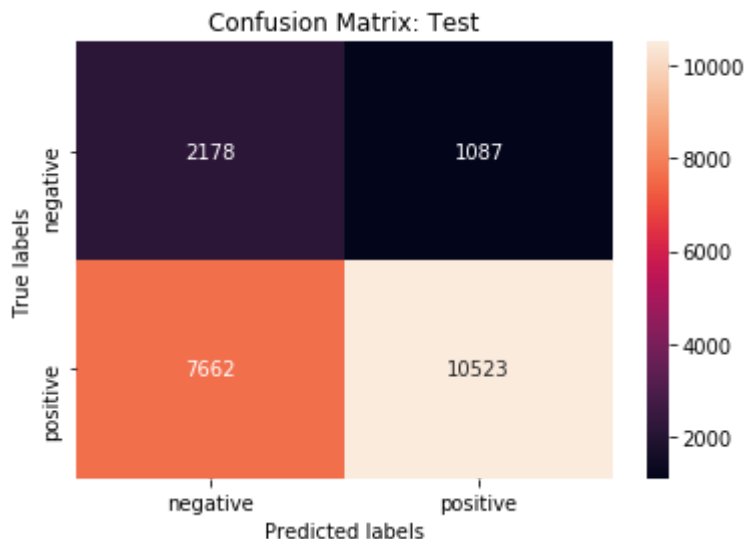
heat_map = plt.subplot()
sn.heatmap(conf_matrix_test, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Test')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])

```

Out[150]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



In [164]:

```

print(X_test.shape)
y_test_new = list(y_test)
X_test_new_avg_w2v = X_test.copy()
X_test_new_avg_w2v['y_actual'] = y_test_new
X_test_new_avg_w2v['y_predicted'] = predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)
print(X_test_new_avg_w2v.shape)

```

(21450, 18)

the maximum value of $tpr \cdot (1 - fpr)$ 0.38601213240975324 for threshold 0.553

(21450, 20)

In [165]:

```

fp_points_avg_w2v = X_test_new_avg_w2v[(X_test_new_avg_w2v['y_actual'] == 0) & (X_test_new_avg_w2v['y_predicted'] == 1)]
fp_points_avg_w2v.shape

```

Out[165]:

(1087, 20)

Word Cloud: Essay

In [166]:

```
comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in fp_points_avg_w2v.cleaned_essay:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

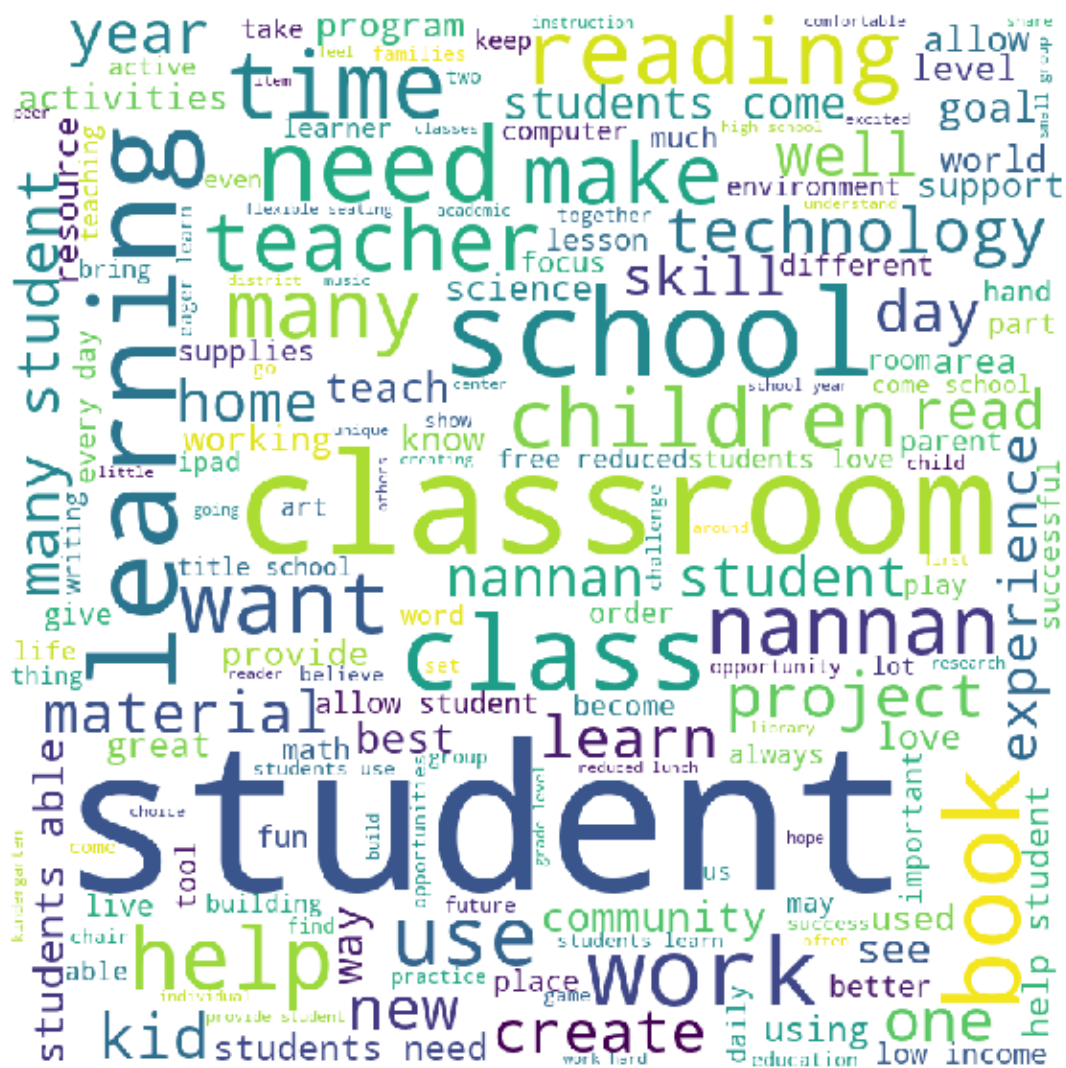
    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800,
                      background_color = 'white',
                      stopwords = stopwords,
                      min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

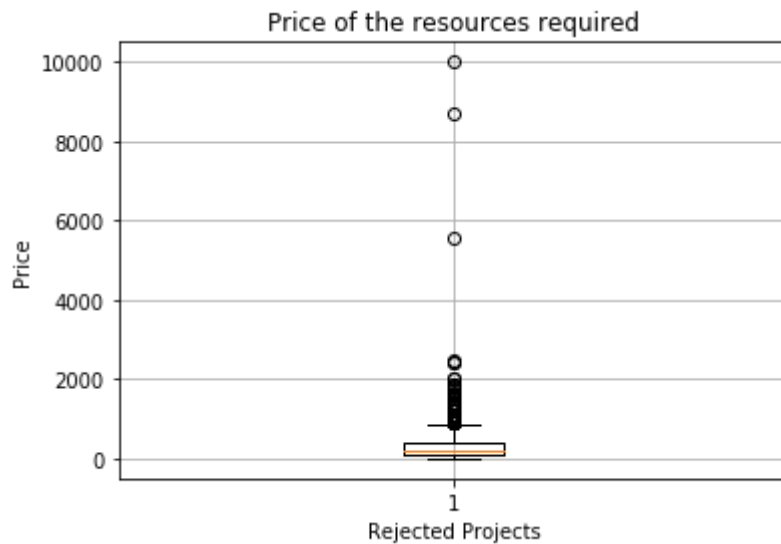
plt.show()
```



Box Plot: Price

In [167]:

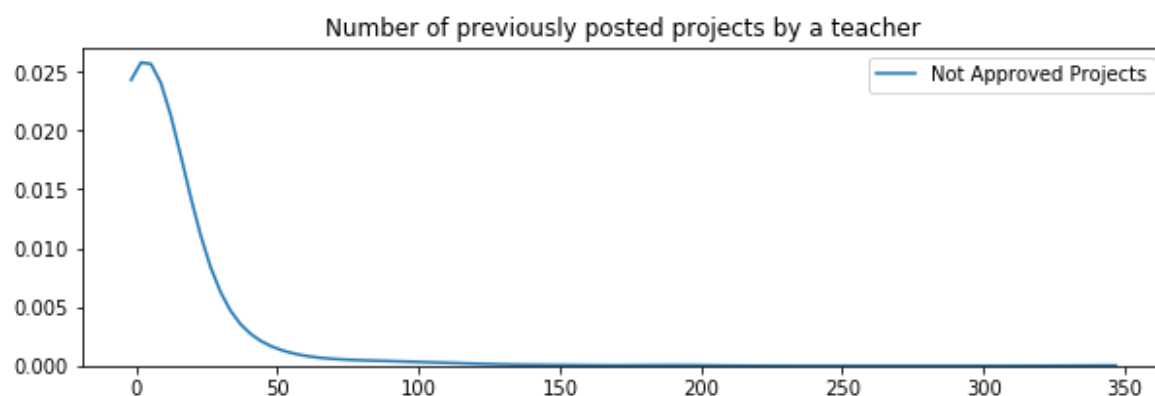
```
plt.boxplot(fp_points_avg_w2v['price'])#[fp_points[fp_points['y_actual']==1]['price'], fp_p
# plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.xlabel('Rejected Projects')
plt.ylabel('Price')
plt.title('Price of the resources required')
plt.grid()
plt.show()
```



PDF: teacher_number_of_previously_posted_projects

In [168]:

```
plt.figure(figsize=(10,3))
# sns.kdeplot(fp_points[fp_points['y_actual']==1]['teacher_number_of_previously_posted_proj
sns.kdeplot(fp_points_avg_w2v['teacher_number_of_previously_posted_projects'],label="Not Ap
plt.title('Number of previously posted projects by a teacher')
plt.legend()
plt.show()
```



2.4.4 Applying Decision Trees on TFIDF W2V, SET 4

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_w2v_feat.pkl","rb")
    X_train_tfidf_w2v_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_w2v_feat.pkl","rb") as
    X_cv_tfidf_w2v_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_w2v_feat.pkl","rb")
    X_test_tfidf_w2v_feat = pickle.load(file)
```

In [130]:

```
X_train_new_tfidf_w2v = vstack([X_train_tfidf_w2v_feat, X_cv_tfidf_w2v_feat])
y_train_new = y_train.append(y_cv)
print(X_train_new_tfidf_w2v.shape)
print(y_train_new.shape)
```

```
(63844, 158)
(63844,)
```

In [0]:

```
grid_search_dt_tfidf_w2v = GridSearchCV(DecisionTreeClassifier(class_weight = "balanced"),
grid_search_dt_tfidf_w2v.fit(X_train_new_tfidf_w2v,y_train_new)
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_tfidf_w2v.pkl",
    pickle.dump(grid_search_dt_tfidf_w2v, file)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_tfidf_w2v.pkl",
    grid_search_dt_tfidf_w2v = pickle.load(file)
```

In [132]:

```
print(grid_search_dt_tfidf_w2v.best_params_)

{'max_depth': 500, 'min_samples_split': 10}
```

In [133]:

```
print(grid_search_dt_tfidf_w2v.best_estimator_)
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
500,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=10,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [134]:

```
test_scores_tfidf_w2v = pd.DataFrame(grid_search_dt_tfidf_w2v.cv_results_['mean_test_score']
test_scores_tfidf_w2v.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                             index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                             inplace = True)
test_scores_tfidf_w2v
```

Out[134]:

	5	10	100	500
1	0.566608	0.566608	0.566608	0.566608
5	0.682445	0.682547	0.682272	0.681701
10	0.784397	0.784406	0.765024	0.733863
50	0.871782	0.873899	0.832742	0.746745
100	0.872384	0.873956	0.833467	0.746790
500	0.873791	0.875273	0.833784	0.746767
1000	0.872383	0.874778	0.833974	0.746763
2000	0.872021	0.874521	0.833355	0.746681

In [135]:

```
train_scores_tfidf_w2v = pd.DataFrame(grid_search_dt_tfidf_w2v.cv_results_['mean_train_score']
train_scores_tfidf_w2v.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                             index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                             inplace = True)
train_scores_tfidf_w2v
```

Out[135]:

	5	10	100	500
1	0.568542	0.568542	0.568542	0.568542
5	0.694880	0.694880	0.694427	0.693314
10	0.849685	0.849089	0.819593	0.772092
50	0.999862	0.999055	0.933713	0.795480
100	0.999898	0.999126	0.933900	0.795501
500	0.999903	0.999119	0.934139	0.795532
1000	0.999900	0.999123	0.934026	0.795501
2000	0.999900	0.999107	0.933860	0.795531

In [136]:

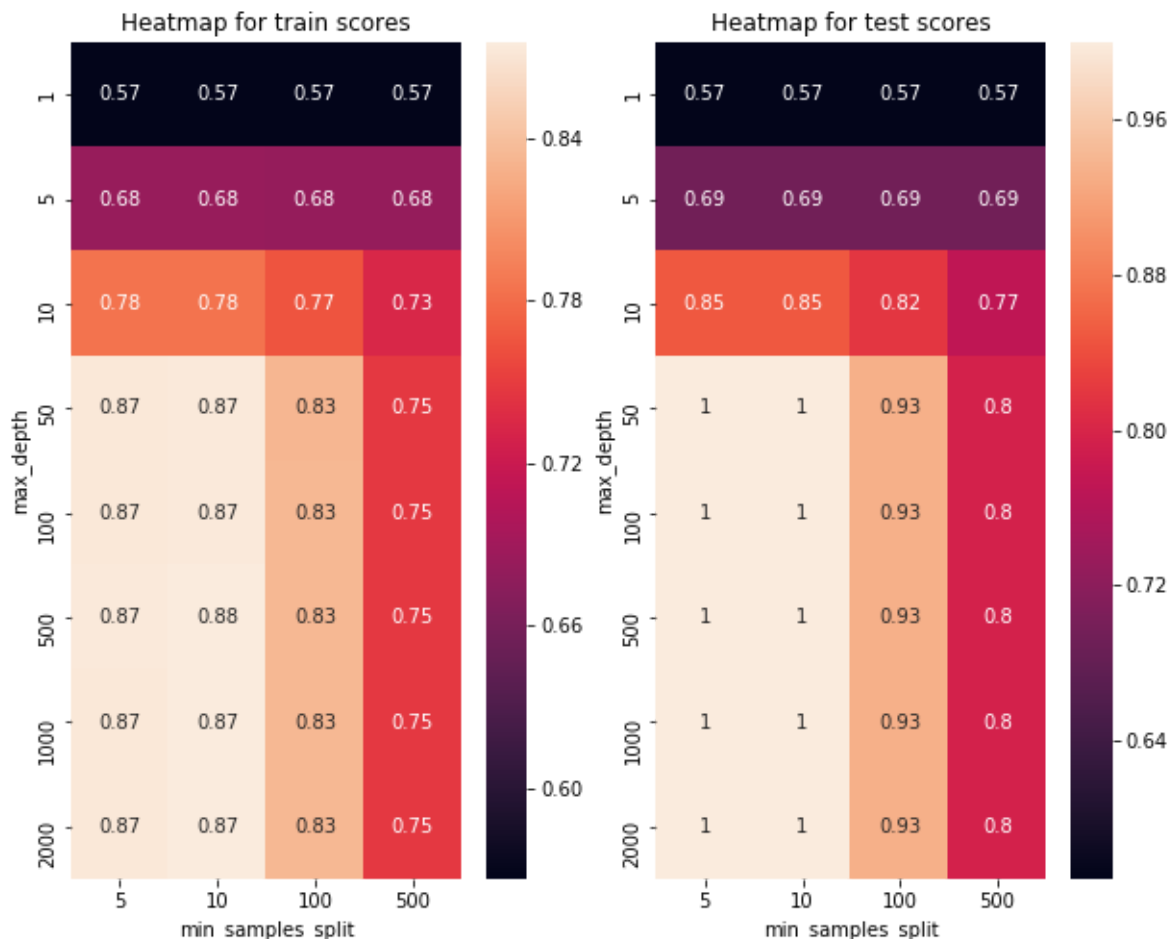
```
train_scores_tfidf_w2v.subtract(test_scores_tfidf_w2v)
```

Out[136]:

	5	10	100	500
1	0.001934	0.001934	0.001934	0.001934
5	0.012434	0.012333	0.012155	0.011613
10	0.065288	0.064683	0.054569	0.038230
50	0.128080	0.125156	0.100972	0.048735
100	0.127514	0.125170	0.100433	0.048711
500	0.126112	0.123846	0.100355	0.048765
1000	0.127517	0.124345	0.100052	0.048738
2000	0.127879	0.124586	0.100505	0.048850

In [137]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 8))
plt.xlabel('max_depth')
sns.heatmap(train_scores_tfidf_w2v, annot=True, xticklabels=param_grid['min_samples_split'],
            yticklabels=param_grid['max_depth'], ax=ax[0])
sns.heatmap(test_scores_tfidf_w2v, annot=True, xticklabels=param_grid['min_samples_split'],
            yticklabels=param_grid['max_depth'], ax=ax[1])
ax[0].set_title("Heatmap for train scores")
ax[1].set_title("Heatmap for test scores")
ax[1].set_xlabel = 'min_samples_split', ylabel = 'max_depth'
fig.show()
```



In [138]:

```
clf_tfidf_w2v_1 = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 50, min_samples_split=500)
clf_tfidf_w2v_1.fit(X_train_new_tfidf_w2v,y_train_new)
```

Out[138]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=50,
```

```
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=500,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')
```

In [0]:

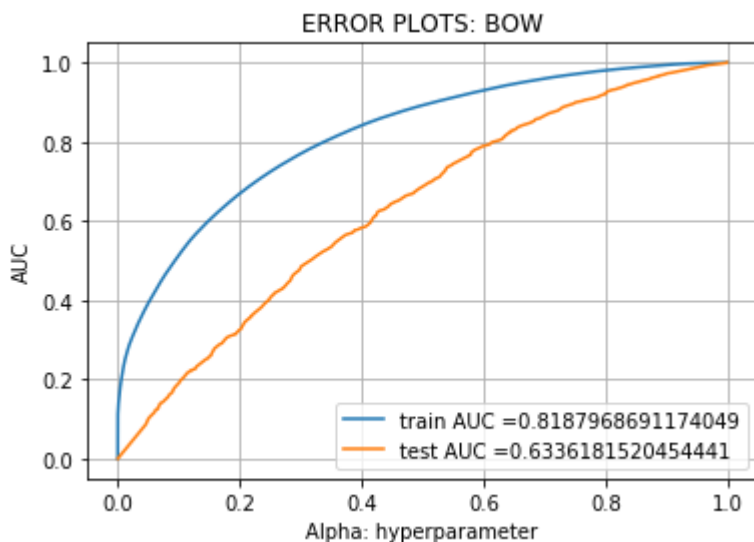
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred_test = clf_tfidf_w2v_1.predict_proba(X_train_new_tfidf_w2v)[:,1]
y_test_pred_test = clf_tfidf_w2v_1.predict_proba(X_test_tfidf_w2v_feat)[:,1]

train_fpr_test, train_tpr_test, tr_thresholds_test = roc_curve(y_train_new, y_train_pred_test)
test_fpr_test, test_tpr_test, te_thresholds_test = roc_curve(y_test, y_test_pred_test)
```

In [140]:

```
plt.plot(train_fpr_test, train_tpr_test, label="train AUC =" + str(auc(train_fpr_test, train_tpr_test)))
plt.plot(test_fpr_test, test_tpr_test, label="test AUC =" + str(auc(test_fpr_test, test_tpr_test)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [141]:

```
clf_tfidf_w2v = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 10, min_samples_leaf=1)
clf_tfidf_w2v.fit(X_train_new_tfidf_w2v, y_train_new)
```

Out[141]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

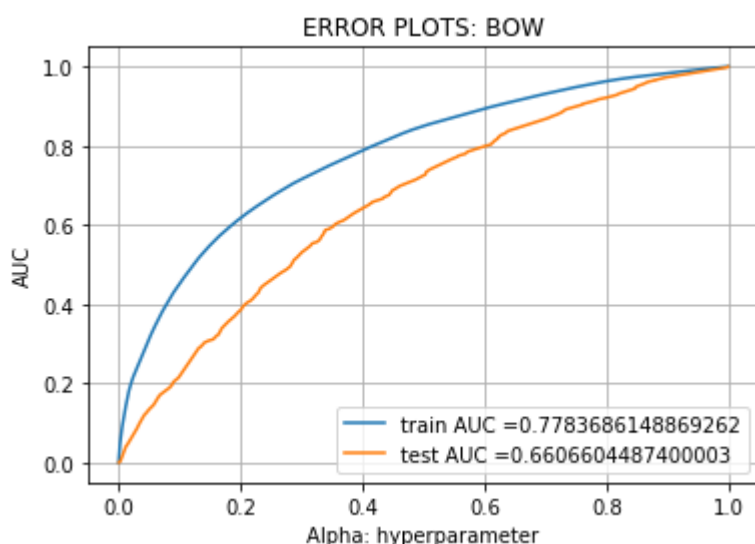
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred = clf_tfidf_w2v.predict_proba(X_train_new_tfidf_w2v)[:,-1]
y_test_pred = clf_tfidf_w2v.predict_proba(X_test_tfidf_w2v_feat)[:,-1]

train_fpr, train_tpr, tr_thresholds= roc_curve(y_train_new, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [152]:

```
plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [0]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix_train = confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
print(confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
conf_matrix_test = confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4986533952638726 for threshold 0.486
the maximum value of tpr*(1-fpr) 0.4986533952638726 for threshold 0.486
[[19669  7255]
 [11719 25201]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.38601213240975324 for threshold 0.553
the maximum value of tpr*(1-fpr) 0.38601213240975324 for threshold 0.553
[[ 2178  1087]
 [ 7662 10523]]
```

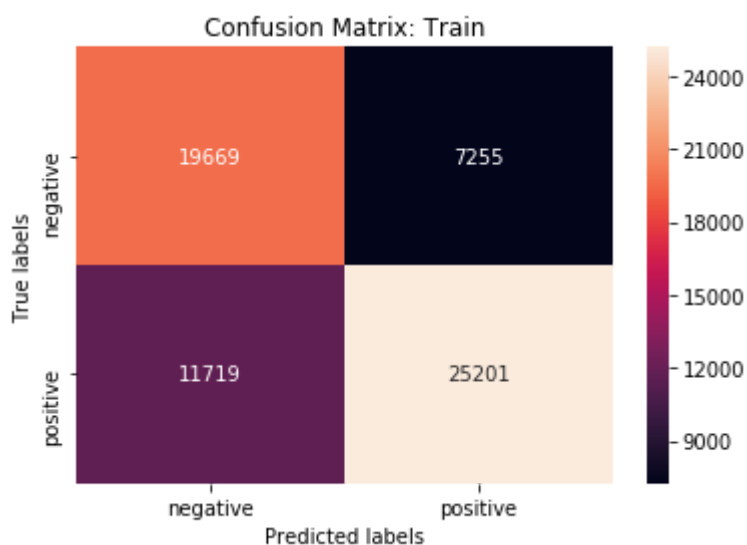
In [154]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix_train, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Train')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[154]:

[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]



In [155]:

```

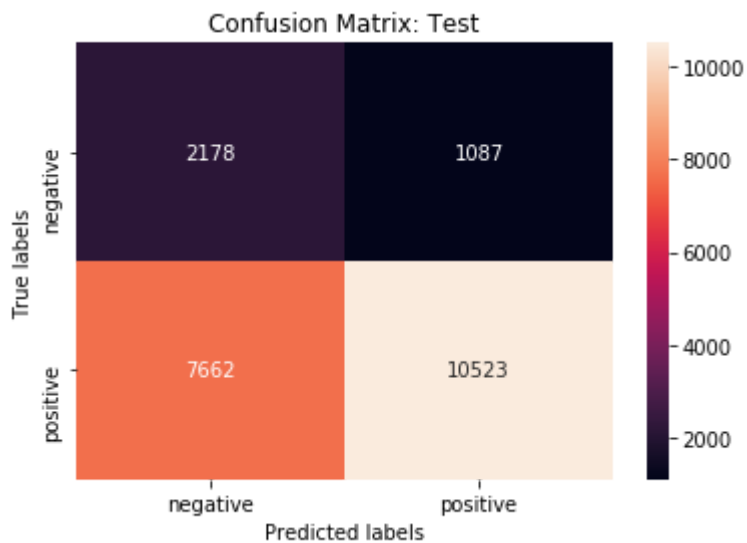
heat_map = plt.subplot()
sn.heatmap(conf_matrix_test, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Test')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])

```

Out[155]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



In [170]:

```

print(X_test.shape)
y_test_new = list(y_test)
X_test_new_tfidf_w2v = X_test.copy()
X_test_new_tfidf_w2v['y_actual'] = y_test_new
X_test_new_tfidf_w2v['y_predicted'] = predict(y_test_pred, tr_thresholds, test_fpr, test_tp)
print(X_test_new_tfidf_w2v.shape)

```

```

(21450, 18)
the maximum value of tpr*(1-fpr) 0.3883545035055312 for threshold 0.543
(21450, 20)

```

In [171]:

```

fp_points_tfidf_w2v = X_test_new_tfidf_w2v[(X_test_new_tfidf_w2v['y_actual'] == 0) & (X_test_new_tfidf_w2v['y_predicted'] == 1)]
fp_points_tfidf_w2v.shape

```

Out[171]:

```
(1160, 20)
```

Word Cloud: Essay

In [172]:

```
comment_words = ' '
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in fp_points_tfidf_w2v.cleaned_essay:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

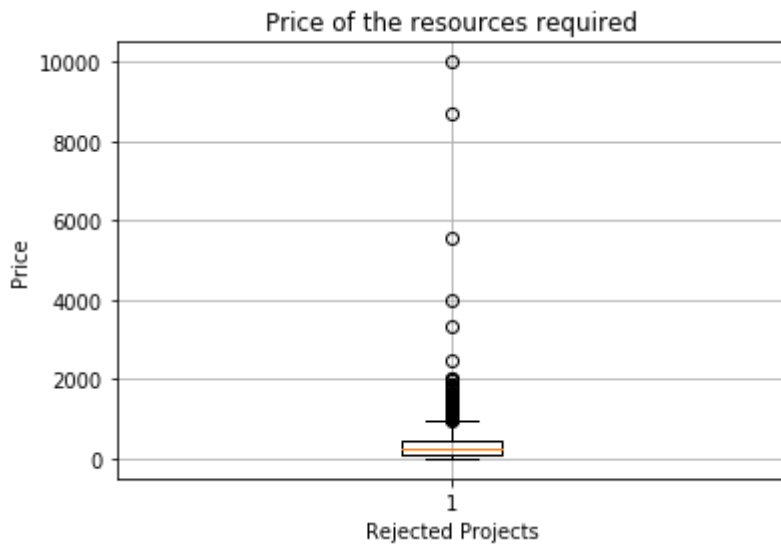
wordcloud = WordCloud(width = 800, height = 800,
                      background_color = 'white',
                      stopwords = stopwords,
                      min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

In [173]:

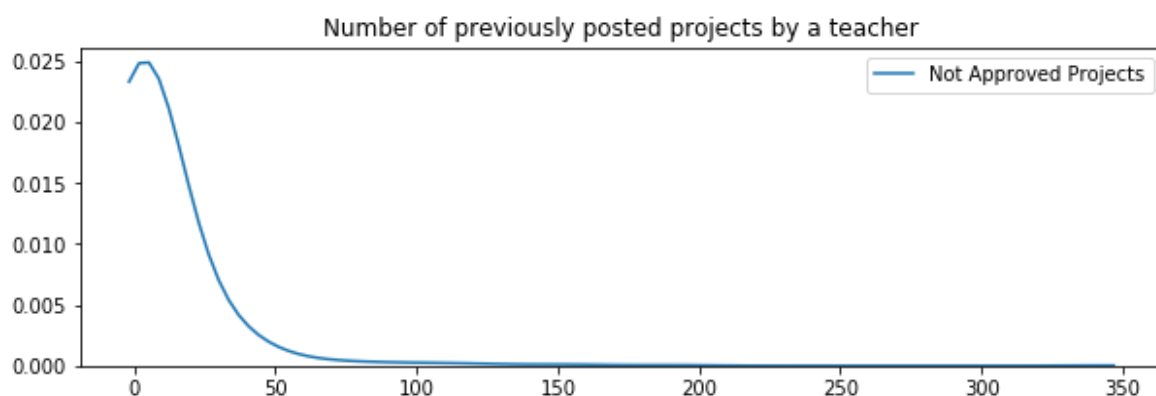
```
plt.boxplot(fp_points_tfidf_w2v['price'])#[fp_points[fp_points['y_actual']==1]['price'], fp
# plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.xlabel('Rejected Projects')
plt.ylabel('Price')
plt.title('Price of the resources required')
plt.grid()
plt.show()
```



PDF: teacher_number_of_previously_posted_projects

In [174]:

```
plt.figure(figsize=(10,3))
# sns.kdeplot(fp_points[fp_points['y_actual']==1]['teacher_number_of_previously_posted_proj
sns.kdeplot(fp_points_tfidf_w2v['teacher_number_of_previously_posted_projects'],label="Not
plt.title('Number of previously posted projects by a teacher')
plt.legend()
plt.show()
```



2.5 [Task-2]Getting top 5k features using feature_importances_

In [0]:

```
with open("/content/drive/My Drive/appliedai/Decision_Trees/dt_tfidf.pkl","wb") as file:  
    pickle.dump(dt_tfidf, file)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Decision_Trees/dt_tfidf.pkl","rb") as file:  
    dt_tfidf = pickle.load(file)
```

In [0]:

```
tfidf_imp_feat = list(clf_tfidf.feature_importances_)  
imp_feat_index = []  
for idx, val in enumerate(tfidf_imp_feat):  
    imp_feat_index.append(idx)  
  
important_feat_df = pd.DataFrame()  
important_feat_df['feat_idx'] = imp_feat_index  
important_feat_df['feat_val'] = tfidf_imp_feat
```

In [176]:

```
important_feat_df.head(2)
```

Out[176]:

	feat_idx	feat_val
0	0	0.0
1	1	0.0

In [177]:

```
important_feat_df.shape
```

Out[177]:

```
(14755, 2)
```

In [178]:

```
important_feat_df.sort_values(by = 'feat_val', ascending = False, inplace = True)
important_feat_df_5k = important_feat_df.head(5000)
important_feat_df_5k.head()
```

Out[178]:

	feat_idx	feat_val
14747	14747	0.185369
14748	14748	0.147411
14746	14746	0.053961
12810	12810	0.049759
4258	4258	0.048607

In [179]:

```
important_feat_df_5k.feat_idx.values
```

Out[179]:

```
array([14747, 14748, 14746, ..., 14565, 14566, 14567])
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_train_tfidf_feat.pkl","rb") as file:
    X_train_tfidf_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_cv_tfidf_feat.pkl","rb") as file:
    X_cv_tfidf_feat = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/X_test_tfidf_feat.pkl","rb") as file:
    X_test_tfidf_feat = pickle.load(file)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_train.pkl","rb") as file:
    y_train = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_cv.pkl","rb") as file:
    y_cv = pickle.load(file)

with open("/content/drive/My Drive/appliedai/SVM/SVM_Dumps/y_test.pkl","rb") as file:
    y_test = pickle.load(file)
```

In [182]:

```
print(X_cv_tfidf_feat.shape)
print(X_test_tfidf_feat.shape)
print(type(y_train))
print(y_cv.shape)
print(y_test.shape)
```

```
(14372, 14755)
(21450, 14755)
<class 'pandas.core.series.Series'>
(14372,)
(21450,)
```

In [0]:

```
from scipy.sparse import coo_matrix, vstack
```

In [184]:

```
X_test_tfidf_feat_new = X_test_tfidf_feat.tocsr()[ :, important_feat_df_5k.feat_idx.values]
print(X_test_tfidf_feat_new.shape)
```

```
(21450, 5000)
```

In [185]:

```
np.unique(y_train_new)
```

Out[185]:

```
array([0, 1])
```

In [186]:

```
X_train_new_tfidf_imp_feat = vstack([X_train_tfidf_feat, X_cv_tfidf_feat])
y_train_new_imp_feat = y_train.append(y_cv)
print(X_train_new_tfidf_imp_feat.shape)
print(y_train_new_imp_feat.shape)
```

```
(63844, 14755)
(63844,)
```

In [187]:

```
X_train_tfidf_feat_new = X_train_new_tfidf_imp_feat.tocsr()[ :, important_feat_df_5k.feat_idx.values]
y_train_new = y_train_new_imp_feat.copy()
print(X_train_tfidf_feat_new.shape)
print(len(y_train_new))
print(type(y_train_new))
```

```
(63844, 5000)
63844
<class 'pandas.core.series.Series'>
```

In [0]:

```
param_grid = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split' : [5, 10, 100]}
```

In [0]:

```
grid_search_dt_tfidf_imp_feat = GridSearchCV(DecisionTreeClassifier(class_weight = "balance
grid_search_dt_tfidf_imp_feat.fit(X_train_tfidf_feat_new,y_train_new)
with open("/content/drive/My Drive/appliedai/Decision_Trees/grid_search_dt_tfidf_imp_feat.p
pickle.dump(grid_search_dt_tfidf_imp_feat, file)
```

In [190]:

```
print(grid_search_dt_tfidf_imp_feat.best_params_)
```

```
{'max_depth': 500, 'min_samples_split': 10}
```

In [191]:

```
print(grid_search_dt_tfidf_imp_feat.best_estimator_)
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
500,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=10,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [192]:

```
test_scores_tfidf_imp_feat = pd.DataFrame(grid_search_dt_tfidf_imp_feat.cv_results_['mean_t
test_scores_tfidf_imp_feat.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                                index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                                inplace = True)
test_scores_tfidf_imp_feat
```

Out[192]:

	5	10	100	500
1	0.566608	0.566608	0.566608	0.566608
5	0.672654	0.672736	0.671906	0.671375
10	0.761670	0.761062	0.746540	0.729366
50	0.862749	0.863850	0.841940	0.796189
100	0.867998	0.871229	0.851774	0.803280
500	0.871846	0.873725	0.853861	0.804181
1000	0.872168	0.872517	0.854135	0.805599

In [196]:

```
train_scores_tfidf_imp_feat = pd.DataFrame(grid_search_dt_tfidf_imp_feat.cv_results_['mean_
train_scores_tfidf_imp_feat.rename(columns = {0: 5, 1: 10, 2: 100, 3: 500},
                                     index = {0: 1, 1: 5, 2: 10, 3: 50, 4: 100, 5: 500, 6: 1000, 7: 2000},
                                     inplace = True)
train_scores_tfidf_imp_feat
```

Out[196]:

	5	10	100	500
1	0.568542	0.568542	0.568542	0.568542
5	0.684630	0.684630	0.683609	0.682921
10	0.807170	0.806691	0.786131	0.757954
50	0.994976	0.993702	0.951449	0.868000
100	0.999294	0.998719	0.965967	0.884729
500	0.999963	0.999648	0.969200	0.890040
1000	0.999961	0.999623	0.969321	0.890970

In [197]:

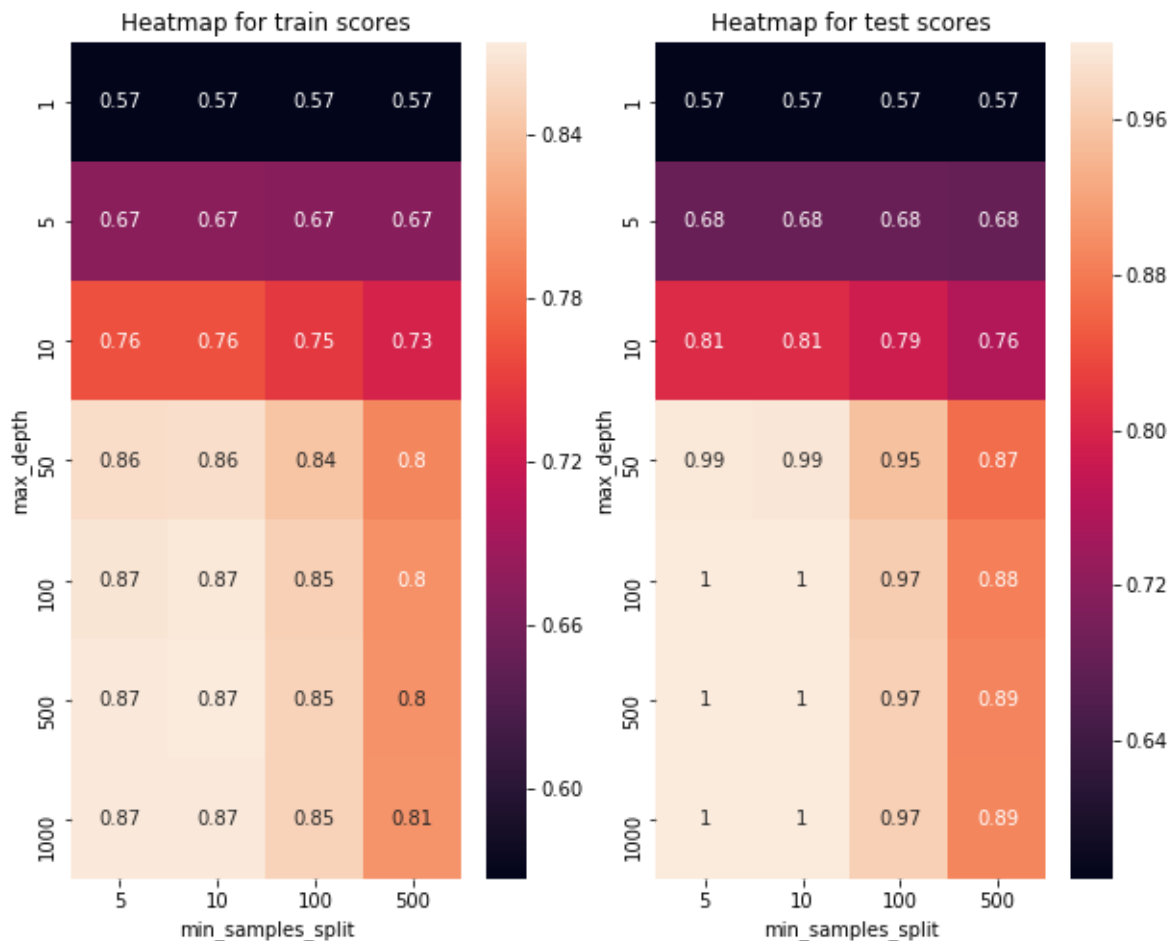
```
train_scores_tfidf_imp_feat.subtract(test_scores_tfidf_imp_feat)
```

Out[197]:

	5	10	100	500
1	0.001934	0.001934	0.001934	0.001934
5	0.011977	0.011895	0.011703	0.011546
10	0.045500	0.045629	0.039592	0.028588
50	0.132227	0.129852	0.109509	0.071811
100	0.131296	0.127491	0.114193	0.081449
500	0.128117	0.125923	0.115339	0.085858
1000	0.127792	0.127107	0.115187	0.085371

In [198]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 8))
plt.xlabel('max_depth')
sns.heatmap(train_scores_tfidf_imp_feat, annot=True, xticklabels=param_grid['min_samples_split'], yticklabels=param_grid['max_depth'])
sns.heatmap(test_scores_tfidf_imp_feat, annot=True, xticklabels=param_grid['min_samples_split'], yticklabels=param_grid['max_depth'])
ax[0].set_title("Heatmap for train scores")
ax[0].set_xlabel = 'min_samples_split', ylabel = 'max_depth'
ax[1].set_title("Heatmap for test scores")
ax[1].set_xlabel = 'min_samples_split', ylabel = 'max_depth'
fig.show()
```



In [199]:

```
clf_tfidf_imp_feat_1 = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 50, mi
clf_tfidf_imp_feat_1.fit(X_train_tfidf_feat_new,y_train_new)
```

Out[199]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
50,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

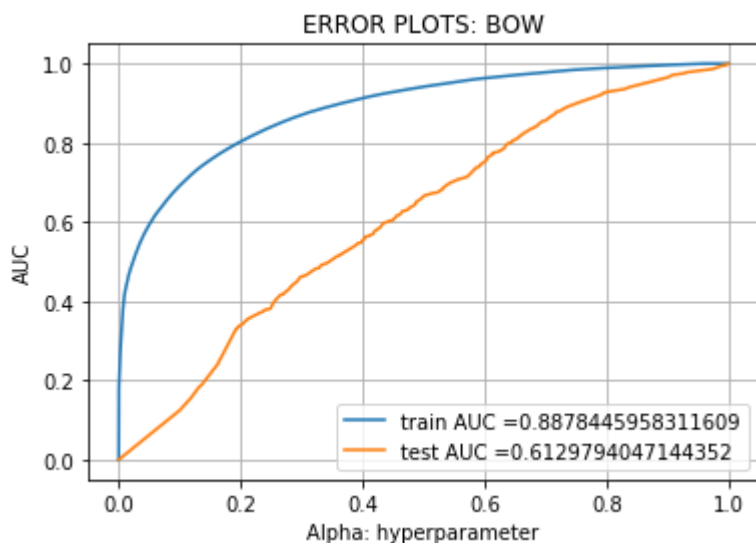
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred_test = clf_tfidf_imp_feat_1.predict_proba(X_train_tfidf_feat_new)[:,1]
y_test_pred_test = clf_tfidf_imp_feat_1.predict_proba(X_test_tfidf_feat_new)[:,1]

train_fpr_test, train_tpr_test, tr_thresholds_test = roc_curve(y_train_new, y_train_pred_te
test_fpr_test, test_tpr_test, te_thresholds_test = roc_curve(y_test, y_test_pred_test)
```

In [201]:

```
plt.plot(train_fpr_test, train_tpr_test, label="train AUC =" +str(auc(train_fpr_test, train_
plt.plot(test_fpr_test, test_tpr_test, label="test AUC =" +str(auc(test_fpr_test, test_tpr_t
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [202]:

```
clf_tfidf_imp_feat = DecisionTreeClassifier(class_weight = 'balanced', max_depth = 10, min_
clf_tfidf_imp_feat.fit(X_train_tfidf_feat_new,y_train_new)
```

Out[202]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=
10,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=500,
                        min_weight_fraction_leaf=0.0, presort=False,
                        random_state=None, splitter='best')
```

In [0]:

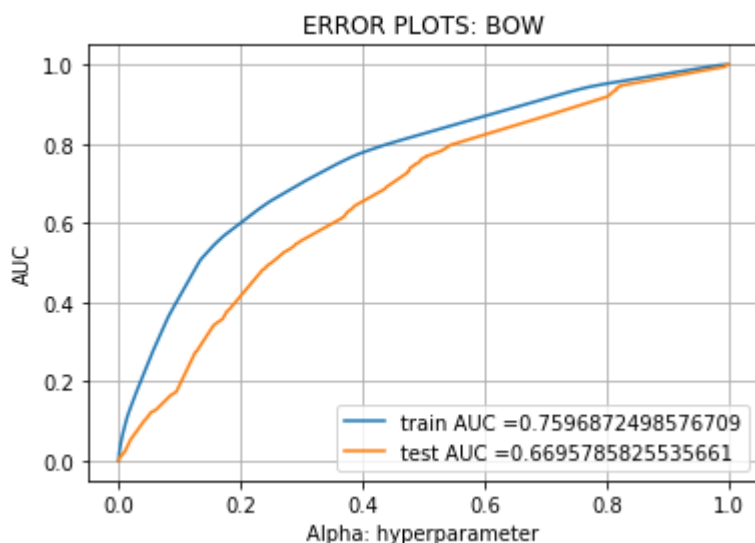
```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
# dt_bow = grid_search_dt_bow.best_estimator_

#predict probabilities for train and test
y_train_pred = clf_tfidf_imp_feat.predict_proba(X_train_tfidf_feat_new)[: ,1]
y_test_pred = clf_tfidf_imp_feat.predict_proba(X_test_tfidf_feat_new)[: ,1]

train_fpr, train_tpr, tr_thresholds= roc_curve(y_train_new, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [205]:

```
plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```

# we are writing our own function for predict, with defined threshould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t, 2))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions

```

In [207]:

```

print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix_train = confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
print(confusion_matrix(y_train_new, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
conf_matrix_test = confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4923798286674437 for threshold 0.476
the maximum value of tpr*(1-fpr) 0.4923798286674437 for threshold 0.476
[[20005  6919]
 [12454 24466]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.39422238933607756 for threshold 0.476
the maximum value of tpr*(1-fpr) 0.39422238933607756 for threshold 0.476
[[ 1998  1267]
 [ 6470 11715]]

```

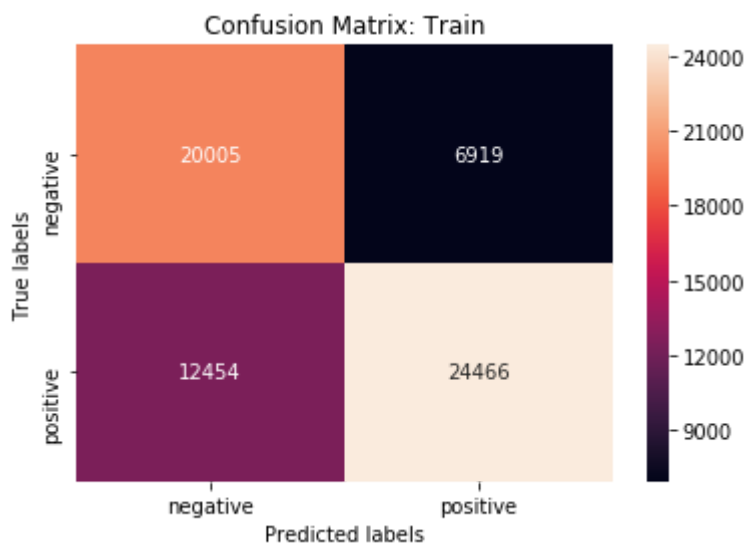
In [208]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix_train, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Train')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[208]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



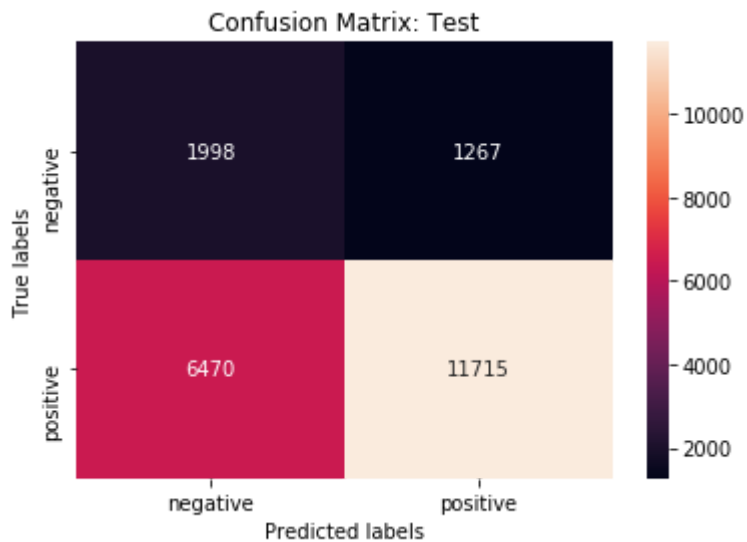
In [209]:

```
heat_map = plt.subplot()
sn.heatmap(conf_matrix_test, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix: Test')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[209]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



3. Conclusion

In [0]:

```
from prettytable import PrettyTable
```

In [0]:

```
dt = PrettyTable()
dt.field_names = ["Vectorizer", "Hyperparameter(max_depth)", "Hyperparameter(min_samples_sp"]
```

In [212]:

```
dt.add_row(["BOW", 10, 500, 0.75])
dt.add_row(["TFIDF", 10, 500, 0.76])
dt.add_row(["Avg W2V", 10, 500, 0.78])
dt.add_row(["TFIDF W2V", 10, 500, 0.78])
dt.add_row(["TFIDF with 5K top features", 10, 500, 0.76])
print(dt)
```

Vectorizer	Hyperparameter(max_depth)	Hyperparameter(min_samples_split)	AUC
BOW	10	50	0.75
TFIDF	10	50	0.76
Avg W2V	10	50	0.78
TFIDF W2V	10	50	0.78
TFIDF with 5K top features	10	50	0.76

In [0]:

```
dt_1 = PrettyTable()
dt_1.field_names = ["Vectorizer", "Hyperparameter(max_depth)", "Hyperparameter(min_samples_split)", "AUC"]
```

In [214]:

```
dt_1.add_row(["BOW", 50, 500, 0.88])
dt_1.add_row(["TFIDF", 50, 500, 0.90])
dt_1.add_row(["Avg W2V", 50, 500, 0.82])
dt_1.add_row(["TFIDF W2V", 50, 500, 0.82])
dt_1.add_row(["TFIDF with 5K top features", 50, 500, 0.89])
print(dt_1)
```

Vectorizer	Hyperparameter(max_depth)	Hyperparameter(min_samples_split)	AUC
BOW	50	50	0.88
TFIDF	50	50	0.9
Avg W2V	50	50	0.82
TFIDF W2V	50	50	0.82
TFIDF with 5K top features	50	50	0.89

